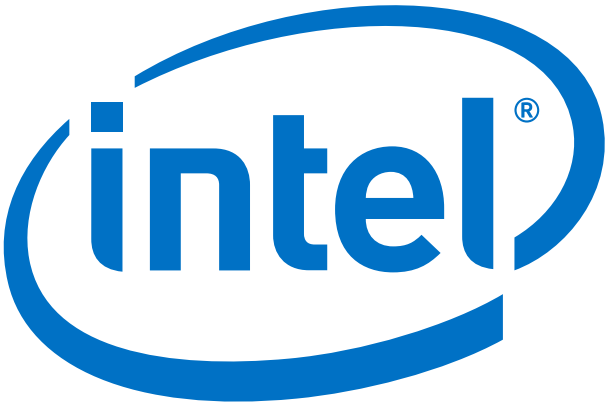


MeteorLake Intel® Firmware Support Package (FSP) Integration Guide

Mon Dec 29 2025 02:16:19

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to describe the steps required to integrate the Intel® Firmware Support Package (FSP) into a boot loader solution. It supports **MeteorLake** platforms with **MeteorLake** processor and **MeteorLake** 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 Related Documents

- *Platform Initialization (PI) Specification v1.7* located at http://www.uefi.org/specifications
- *Intel® Firmware Support Package: External Architecture Specification (EAS) v2.1* located at https://cdrdv2.intel.com/v1/dl/getContent/611786
- *Boot Setting File Specification (BSF) v1.0* https://firmware.intel.com/sites/default/files/BSF_1_0.pdf
- *Binary Configuration Tool for Intel® Firmware Support Package* available at http://www.intel.com/fsp

1.4 Acronyms and Terminology

Acronym	Definition
BCT	Binary Configuration Tool
BDSM	Base Data Of Stolen Memory
BSF	Boot Setting File
BSP	Boot Strap Processor

Acronym	Definition
BWG	BIOS Writer's Guide
CAR	Cache As Ram
CRB	Customer Reference Board
DPR	DMA Protected Range
FIT	Firmware Interface Table
FSP	Firmware Support Package
FSP API	Firmware Support Package Interface
FW	Firmware
GTT	Graphics Translation Table
IED	Intel Enhanced Debug
IFWI	Integrated Firmware Image
IOT	Internal Observation Trace
MRC	Memory Reference Code (Memory Init code encapsulated by FSP-M)
MOT	Memory Observation Trace
PCH	Platform Controller Hub
PMC	Power Management Controller
PMRR	Protected Memory Range Reporting
REMAP	Remapped Memory Area
RVP	Reference and Validation Platform
SBSP	System BSP
SMI	System Management Interrupt
SMM	System Management Mode
SMRAM	System Management Mode RAM
SPI	Serial Peripheral Interface
TOLUD	Top of Low Usable Memory
TOUUD	Top of Upper Usable Memory
TSEG	Memory Reserved at the Top of Memory to be used as SMRAM
UPD	Updatable Product Data

Chapter 2

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 MinPlatform, Intel® Slim Bootloader, 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 tools provided by Intel. This step is optional as configuration data may also be provided at runtime.
- **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 FSP distribution package contains the following:
  - FSP Binary
  - FSP Integration Guide
  - BSF Configuration File
  - Data Structure Header Files
- 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)**
  - FSP_Integration_Guide.pdf
  - FSP_Integration_Guide.chm
- **Include**
  - FsptUpd.h, FspmUpd.h and FspsUpd.h (FSP UPD structure and related definitions)
  - GpioSampleDef.h (Sample enum definitions for GPIO table)
- FspBinPkg.dec (EDKII declaration file for package)
- Fsp.bsf (BSF file for configuring the data using BCT tool)
- Fsp.fd (FSP Binary)

Chapter 3

FSP Integration

3.1 Assumptions Used in this Document

The FSP for this 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) or SplitFspBin.py before integrating to the boot loader.

For other assumptions and conventions, please refer to Chapter 8 and 9 of the FSP External Architecture Specification version 2.2.

3.2 Boot Flow

Please refer Chapter 7 in the FSP External Architecture Specification version 2.2 for Boot flow chart. The FSP for this platform supports dispatch mode, see Chapter 7 and 9 of the FSP External Architecture Specification version 2.2 for a description of dispatch mode.

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 2.2 with a HeaderRevision of 5.

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 FS↩ P-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:0x0A001460).

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 Memory Map

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

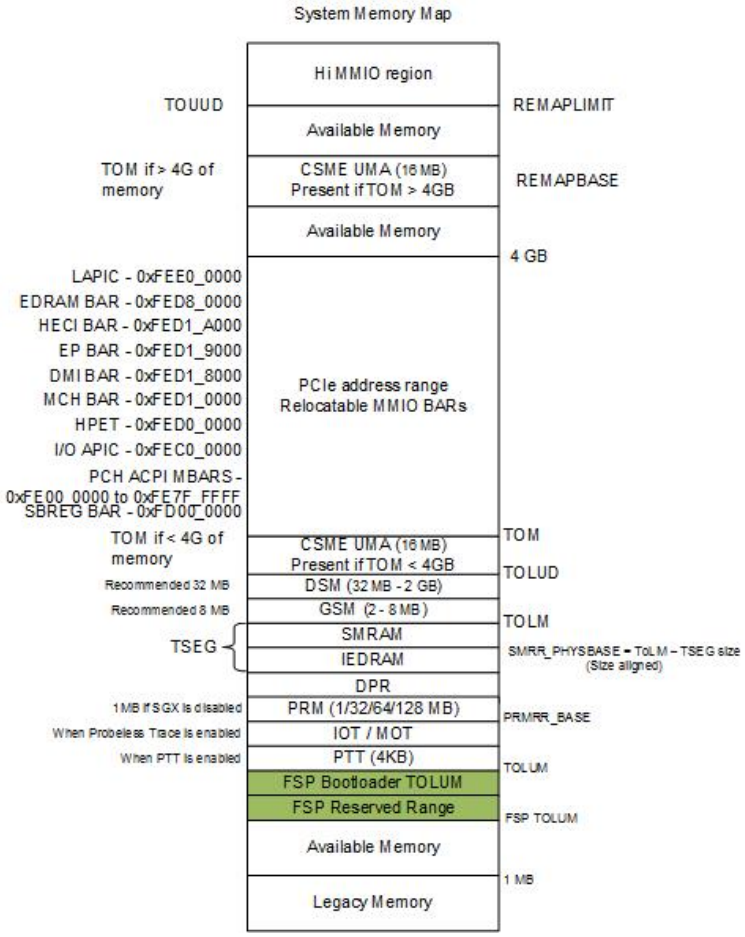


Figure 3.1 System Memory Map

3.7 Optional modules

For products using FSP in API mode, having SPI size constraints, the following modules can be safely removed to help save additional space. These modules are included in FSP binary primarily when FSP is used in dispatch mode:

- PchInitDxeFsp

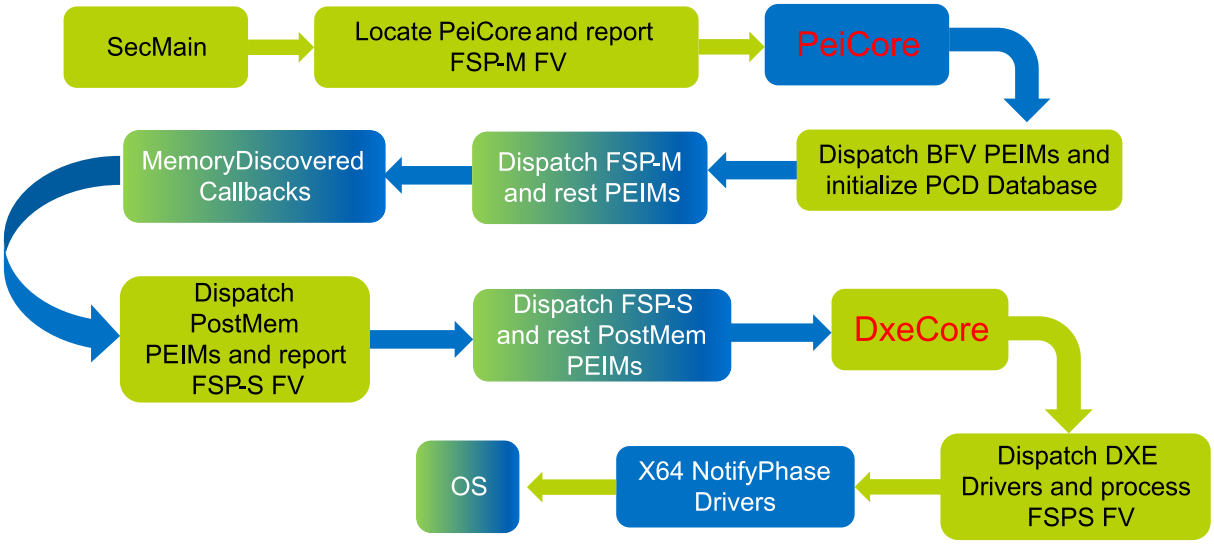
- HeciInitDxeFsp
- SiInitDxeFsp
- MtlPchInitDxeFsp
- CpuMpPei
- FspS3Notify
- FspEndOfPei2Peim
- GpioV2PpiInitPei
- PchInitFsp
- HeciInitFsp

Chapter 4

FSP Dispatch Mode

Overview

The FSP for this platform supports dispatch mode. Support for dispatch mode can be detected by checking if FSP_INFO_HEADER->ImageAttribute[BIT1] == 1. Dispatch mode is intended to enable FSP to integrate well in to UEFI bootloader implementations. Dispatch mode implements a boot flow that is as close to a standard UEFI boot flow as possible. In dispatch mode, the FSP is exposed as standard Firmware Volumes (FVs) directly to the bootloader. The PEIMs in these FVs are executed in the same PEI environment as the boot loader. In dispatch mode, the PPI database, PCD database, and HOB list are shared between the boot loader and the FSP.



Blue blocks are from the FSP binary and green blocks are from the bootloader. Blocks with mixed colors indicate that both bootloader and FSP modules are dispatched during that phase of the boot flow. In dispatch mode, the Notify↩ Phase() API is not used. Instead, FSP-S contains DXE drivers that implement the native callbacks on equivalent events for each of the NotifyPhase() invocations.

In dispatch mode, the the PPI database and PCD database are used for providing policy data from the bootloader to FSP. Because these mechanisms provide a great deal of flexibility, dispatch mode does not constrain the method for passing policy data as strongly as API mode. The following sections describe the dispatch mode policy initialization flow used specifically for this platform.

4.1 Dispatch Mode Policy Init

For the plurality of platform designs, most of the policy options provided by FSP do not need to be modified by the bootloader.

To ease this common case, policy initialization has been broken in to two phases: 1. Policy Init and 2. Policy Update.

Policy Init creates the policy data structures with all default policy values pre-populated. Policy Init is implemented using the `factory method pattern`. The FSP provides an API to the bootloader for constructing the policy data structures. Because the factory method is provided by the FSP, backwards compatibility is made substantially easier. The FSP can be assured that the policy data structures match the definitions used at the time the FSP was compiled. As long as new fields are added to the bottom of policy structures, the bootloader may continue to use an older version of the policy data structures at compile time and retain compatibility with both new and old FSP binaries.

There are two factory methods, one for FSP-M and one for FSP-S. PeiPreMemPolicyInit() is provided by FSP-M, PeiPolicyInit() is provided by FSP-S. These methods are exposed to the bootloader using PPIs, PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI for FSP-M and PEI_SI_DEFAULT_POLICY_INIT_PPI for FSP-S. The usage of PPIs ensures ABI stability between the FSP and the bootloader.

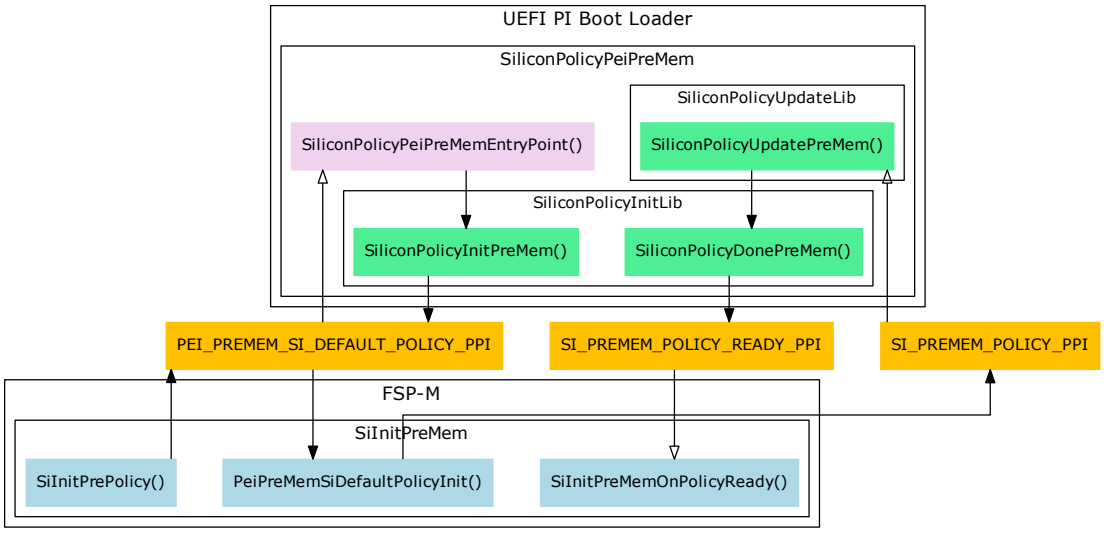
While Policy Init is implemented by the FSP, Policy Update is implemented by the bootloader. Policy Update uses a read-modify-write operation to apply any motherboard specific settings to the policy data while leaving the default values intact when appropriate. After Policy Update is done, FSP may run its SOC initialization code.

4.1.1 FSP-M Policy Initialization Flow

The follow graph shows the policy initialization flow for FSP-M.

Note

The hollow arrow heads in the flow diagram below indicate that action by the PEI Foundation is required for the flow to proceed.



Note

The function names and PEIMs used by the UEFI PI boot loader are implementation dependent and may vary from those shown here. The function names and PEIMs used by MinPlatform are provided as an example.

Execution of FSP-M begins after FspmWrapperInit() in IntelFsp2WrapperPkg/FspmWrapperPeim/FspmWrapperPeim.c installs a EFI_PEI_FIRMWARE_VOLUME_INFO_PPI instance for FSP-M. This causes the PEI foundation to become aware of the FV(s) in FSP-M, which schedules all PEIMs in FSP-M for dispatch by PEI. This results in the SiInitPreMem PEIM in FSP-M being executed.

One of the actions performed by the SiInitPreMem entry-point is calling the SiInitPrePolicy() function, which installs the PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI. The DEPEX for MinPlatformPkg/PlatformInit/SiliconPolicyPei/SiliconPolicyPeiPreMem.inf requires PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI to exist in the PPI database before SiliconPolicyPeiPreMem can run. While SiliconPolicyPeiPreMem.inf does not directly add PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI to the DEPEX, SiliconPolicyPeiPreMem.inf does statically link against MeteorlakeSiliconPkg/Library/PeiSiliconPolicyInitLib/PeiPreMemSiliconPolicyInitLib.inf, which adds PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI to the DEPEX. After the SiInitPreMem entry-point completes the dependency will be satisfied, making SiliconPolicyPeiPreMem eligible for dispatch.

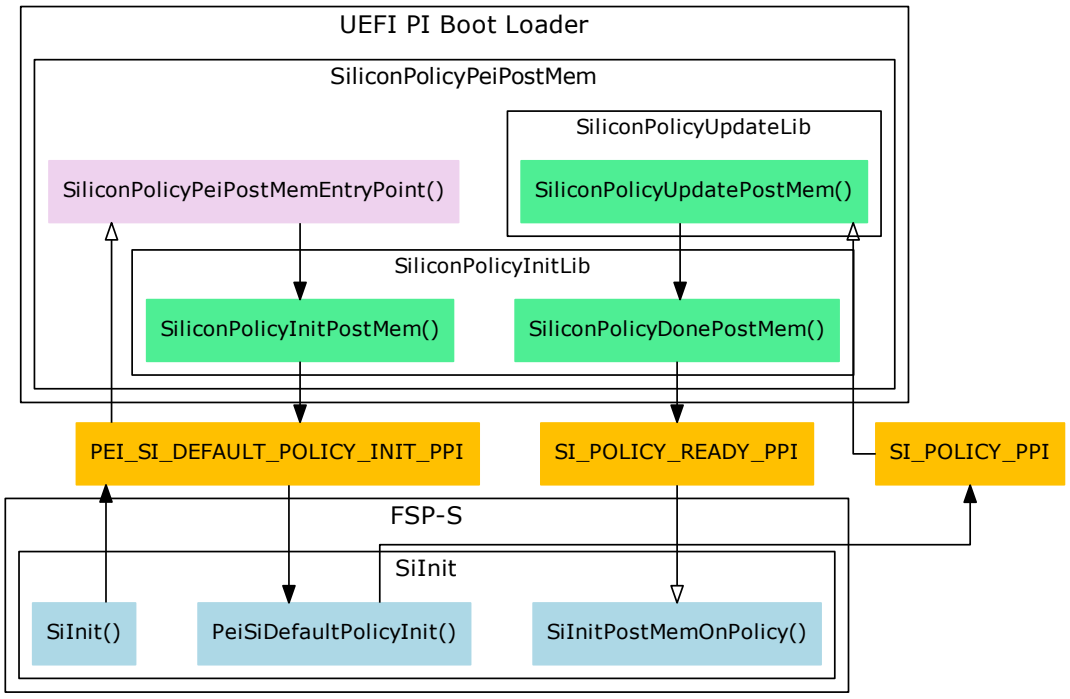
Next SiliconPolicyPeiPreMem PEIM will run. SiliconPolicyPeiPreMemEntryPoint() in MinPlatformPkg/PlatformInit/SiliconPolicyPei/SiliconPolicyPeiPreMem.c will call SiliconPolicyInitPreMem() in MeteorlakeSiliconPkg/Library/PeiSiliconPolicyInitLib/PeiPolicyInitPreMem.c. This function will locate the instance of PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI previously created by SiInitPreMem and use it to call PeiPreMemSiDefaultPolicyInit() in SiInitPreMem. PeiPreMemSiDefaultPolicyInit() will construct SI_PREMEM_POLICY_PPI, which contains all the policy data needed by FSP-M. SiliconPolicyInitPreMem() will then locate SI_PREMEM_POLICY_PPI and return it to SiliconPolicyPeiPreMemEntryPoint().

Next, SiliconPolicyPeiPreMemEntryPoint() takes the pointer to the policy data returned by SiliconPolicyInitPreMem() and passes it to SiliconPolicyUpdatePreMem(). SiliconPolicyUpdatePreMem() is part of SiliconPolicyUpdateLib, which is statically linked to SiliconPolicyPeiPreMem. SiliconPolicyUpdateLib is special since it is only piece of motherboard specific code in this flow. An example of SiliconPolicyUpdatePreMem() is seen in MeteorlakeOpenBoardPkg/MeteorlakeMRvp/Policy/Library/PeiSiliconPolicyUpdateLib/PeiSiliconPolicyUpdateLib.c. SiliconPolicyUpdatePreMem() uses a read-modify-write operation to apply any motherboard specific settings to the policy data while leaving the default values intact when appropriate.

After SiliconPolicyUpdatePreMem() applies any motherboard specific updates, SiliconPolicyPeiPreMemEntryPoint() calls SiliconPolicyDonePreMem() in MeteorlakeSiliconPkg/Library/PeiSiliconPolicyInitLib/PeiPolicyInitPreMem.c. SiliconPolicyDonePreMem() dumps the policy data to the debug log (on a DEBUG build of MinPlatform only) and installs the SI_PREMEM_POLICY_READY_PPI. SI_PREMEM_POLICY_READY_PPI tells FSP-M that everything is ready and that it can run the SOC init code now. Installing SI_PREMEM_POLICY_READY_PPI causes the PEI Foundation to signal a PeiServices->NotifyPpi() callback previously set up by SiInitPreMem. This callback invokes SiInitPreMemOnPolicyReady() in SiInitPreMem, which runs MRC and all the other SOC init code in FSP-M.

4.1.1 FSP-S Policy Initialization Flow

Policy Initialization for FSP-S follows essentially the same flow as FSP-M. The primary difference is function and PPI names do not include the "PreMem" qualifier.



4.2 Config Blocks

Overview

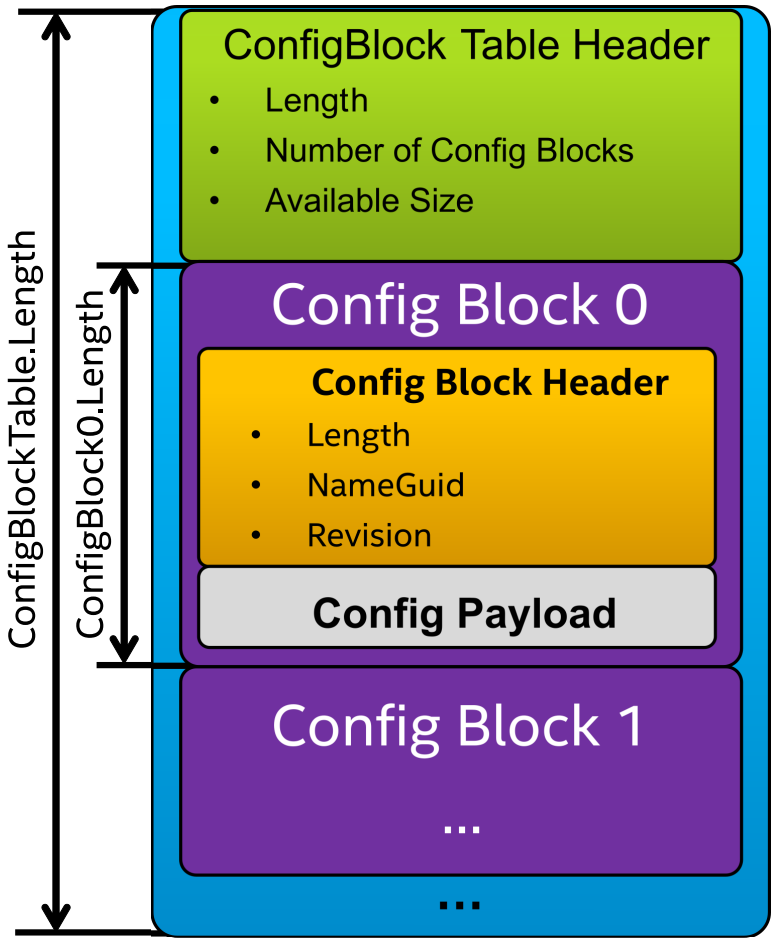
Config Blocks are a binary data serialization format with several unique properties that are useful for firmware implementations.

The serialization format is simple to implement in the C language, minimizing the code needed to implement it; this minimizes the size of the FSP binary. Second, the format is binary backwards compatible; enabling the bootloader to use an older version of the data structure at compile time and retain compatibility with both new and old FSP binaries. Third, the format is mutable in memory without requiring the data to be re-serialized. Finally, the format is position independent with no need for a base address or a rebase operation. This allows the data structure to be moved from pre-memory to post-memory without applying any fixups.

Data in SI_PREMEM_POLICY_PPI and SI_POLICY_PPI are stored using the Config Block format.

4.2.1 Config Block Data Format

In essence, Config Blocks are multiple C structures that are stored in a single continuous memory range using run length encoding.



The overall structure is termed a Config Block Table. The Config Block Table starts with a CONFIG_BLOCK_TABLE_HEADER structure. This header indicates the number of Config Blocks in the Config Block Table, the total size of the table, and the amount of free space remaining in the table. The table may be over-provisioned, which allows additional Config Blocks to be added after the table's initial creation.

The size of all Config Blocks must be an even multiple of 32-bits as all Config Blocks are DWORD aligned. When enumerating a Config Block Table one may assume that all Config Blocks are packed back-to-back, hence the enumeration algorithm can use the length of each config block to find the next config block in the table.

Each Config Block has a GUID that identifies which C structure the Config Block contains. All Config Blocks start with a standardized CONFIG_BLOCK_HEADER, which allows all the structures in the table to be enumerated even if the format of the data payload inside each Config Block is unknown.

```
typedef struct _CONFIG_BLOCK_HEADER {
  EFI_HOB_GUID_TYPE GuidHob;                    ///< Offset 0-23  GUID extension HOB header
  UINT8             Revision;                   ///< Offset 24    Revision of this config block
  UINT8             Attributes;                 ///< Offset 25    The main revision for config block
  UINT8             Reserved[2];                ///< Offset 26-27 Reserved for future use
} CONFIG_BLOCK_HEADER;
```

The CONFIG_BLOCK_HEADER uses the header of a GUID HOB to store its length and GUID. This makes it easy to copy Config Blocks to the HOB list. Copying Config Blocks to the HOB list is a common use case. Often the ACPI code uses policies that were initially supplied by a Config Block. Copying the Config Block to the HOB list makes it easy for DXE phase to copy any needed policies to the ACPI global NVS memory later.

The Revision field assists with binary backward compatibility. Whenever a new field is added to a Config Block, the new field is added to the bottom of the Config Block and the Revision is incremented by 1. Thus if a bootloader wishes to retain compatibility with new and old versions of the FSP binary, it may do so by either by only using fields that are present in Revision 1 of the Config Block or by checking the Revision number before modifying fields added since Revision 1.

The CONFIG_BLOCK_TABLE_HEADER builds upon the CONFIG_BLOCK_HEADER:

```
typedef struct _CONFIG_BLOCK_TABLE_STRUCT {
  CONFIG_BLOCK_HEADER           Header;          ///< Offset 0-27  GUID number for main entry of config
      block
  UINT8                         Rsvd0[2];        ///< Offset 28-29 Reserved for future use
  UINT16                        NumberOfBlocks;  ///< Offset 30-31 Number of config blocks (N)
  UINT32                        AvailableSize;   ///< Offset 32-35 Current config block table size
///
/// Individual Config Block Structures are added starting here
///
} CONFIG_BLOCK_TABLE_HEADER;
```

4.2.2 Config Block Library APIs

To assist in the creation and parsing of Config Blocks, Intel has provided an open source Config Block library in `IntelSiliconPkg/Library/BaseConfigBlockLib`. This library provides the following functions:

- GetConfigBlock()
- AddConfigBlock()
- CreateConfigBlockTable()

In most cases, bootloaders will only need to call GetConfigBlock().

4.2.3 Config Blocks used by FSP-M

On **MeteorLake** platforms SI_PREMEM_POLICY_PPI contains a Config Block Table. PEI_PREMEM_SI_DEFAULT_POLICY_INIT_P will construct this Config Block Table with all the below Config Blocks pre-populated. Additionally, all the Config Blocks will be initialized with default policy values as described in section 4.1.

SI_PREMEM_POLICY_PPI contains the following Config Blocks:

Todo This list is from TigerLake, needs to be updated for MeteorLake

- SI_PREMEM_CONFIG
- HOST_BRIDGE_PREMEM_CONFIG
- CPU_INIT_PREMEM_CONFIG
- CPU_DMI_PREMEM_CONFIG
- CPU_PCIE_RP_PREMEM_CONFIG
- CPU_SECURITY_PREMEM_CONFIG
- CPU_TRACE_HUB_PREMEM_CONFIG
- CPU_TXT_PREMEM_CONFIG
- GRAPHICS_PEI_PREMEM_CONFIG
- HDAUDIO_PREMEM_CONFIG
- HYBRID_GRAPHICS_CONFIG
- IPU_PREMEM_CONFIG

- ISH_PREMEM_CONFIG
- ME_PEI_PREMEM_CONFIG
- MEMORY_CONFIG_NO_CRC
- MEMORY_CONFIGURATION
- OVERCLOCKING_PREMEM_CONFIG
- PCH_DCI_PREMEM_CONFIG
- PCH_GENERAL_PREMEM_CONFIG
- PCH_HSIO_PCIE_PREMEM_CONFIG
- PCH_HSIO_SATA_PREMEM_CONFIG
- PCH_LPC_PREMEM_CONFIG
- PCH_PCIE_RP_PREMEM_CONFIG
- PCH_SMBUS_PREMEM_CONFIG
- PCH_WDT_PREMEM_CONFIG
- PCIE_PEI_PREMEM_CONFIG
- PCIE_PREMEM_CONFIG
- PRAM_PREMEM_CONFIG
- SA_MISC_PEI_PREMEM_CONFIG
- TCSS_PEI_PREMEM_CONFIG
- TELEMETRY_PEI_PREMEM_CONFIG
- TWOLM_PREMEM_CONFIG
- VTD_CONFIG

4.2.4 Config Blocks used by FSP-S

On **MeteorLake** platforms SI_POLICY_PPI contains a Config Block Table. PEI_SI_DEFAULT_POLICY_INIT_PPI will construct this Config Block Table with all the below Config Blocks pre-populated. Additionally, all the Config Blocks will be initialized with default policy values as described in section 4.1.

SI_POLICY_PPI contains the following Config Blocks:

Todo This list is from TigerLake, needs to be updated for MeteorLake

- SI_CONFIG
- HOST_BRIDGE_PEI_CONFIG
- ADR_CONFIG
- AMT_PEI_CONFIG
- BIOS_GUARD_CONFIG
- CNVI_CONFIG

- CPU_INIT_CONFIG
- CPU_PCIE_CONFIG
- CPU_PID_TEST_CONFIG
- CPU_POWER_MGMT_BASIC_CONFIG
- CPU_POWER_MGMT_CUSTOM_CONFIG
- CPU_POWER_DELIVERY_CONFIG
- CPU_POWER_MGMT_TEST_CONFIG
- CPU_POWER_MGMT_VR_CONFIG
- CPU_TEST_CONFIG
- GBE_CONFIG
- GNA_CONFIG
- GRAPHICS_PEI_CONFIG
- HDAUDIO_CONFIG
- HYBRID_STORAGE_CONFIG
- IEH_CONFIG
- ISH_CONFIG
- ME_PEI_CONFIG
- PCH_DMI_CONFIG
- PCH_ESPI_CONFIG
- PCH_FIVR_CONFIG
- PCH_FLASH_PROTECTION_CONFIG
- PCH_GENERAL_CONFIG
- PCH_HSIO_CONFIG
- PCH_INTERRUPT_CONFIG
- PCH_IOAPIC_CONFIG
- PCH_LOCK_DOWN_CONFIG
- PCH_P2SB_CONFIG
- PCH_PCIE_CONFIG
- PCH_PM_CONFIG
- PEI_ITBT_CONFIG
- PSF_CONFIG
- RST_CONFIG
- RTC_CONFIG
- SA_MISC_PEI_CONFIG
- SATA_CONFIG
- SERIAL_IO_CONFIG

- TCSS_PEI_CONFIG
- TELEMETRY_PEI_CONFIG
- THC_CONFIG
- THERMAL_CONFIG
- TSN_CONFIG
- USB_CONFIG
- USB2_PHY_CONFIG
- USB3_HSIO_CONFIG
- VMD_PEI_CONFIG

4.3 FSP Error Information

In the case of a fatal error occurring during the execution of the FSP, it may not be possible for the FSP to continue.

If a fatal error that prevents the successful completion of the FSP occurs, the FSP may use FSP_ERROR_INFO to report this error to the bootloader. During PEI phase, (*PeiServices)->ReportStatusCode() shall be used to transmit this error notification to the bootloader. During DXE phase, EFI_STATUS_CODE_PROTOCOL.Report↩StatusCode() shall be used to transmit this error notification to the bootloader. The bootloader must ensure that ReportStatusCode() services are available before FSP-M begins execution.

```
typedef struct {
  EFI_STATUS_CODE_DATA    DataHeader;
  EFI_GUID                ErrorType;
  EFI_STATUS              Status;
} FSP_ERROR_INFO;
```

FSP_ERROR_INFO is provided as the optional EFI_STATUS_CODE_DATA parameter to ReportStatusCode(). EFI_STATUS_CODE_DATA provides a CallerId GUID, this CallerId combined with the ErrorType GUID describes the error to the bootloader. The FSP for this platform implements the following CallerId GUIDs:

CallerId	Description
{0x1f4dc7e9, 0x26ca, 0x4336, {0x8c, 0xe3, 0x39, 0x31, 0x03, 0xb5, 0xf3, 0xd7}}	ME
{0x98230916, 0xe632, 0x49ff, {0x81, 0x81, 0x55, 0xce, 0xe5, 0x10, 0x36, 0x89}}	System Agent
{0x5a47c211, 0x642f, 0x4f92, {0x9c, 0xb3, 0x7f, 0xeb, 0x93, 0xda, 0xdd, 0xba}}	MRC

If the CallerId parameter is not a GUID in the table above, then it should be the GUID that identifies the PEIM or DXE driver which was executing at the time of the error.

The following ErrorType GUIDs are implemented:

ErrorType	Description
{0x948585c4, 0x76a4, 0x45bb, {0xbe, 0x6c, 0x39, 0x61, 0xc3, 0xab, 0xde, 0x15}}	ME EOP failure
{0x8106a5cc, 0x30ba, 0x41cf, {0xa1, 0x78, 0x63, 0x38, 0x91, 0x11, 0xae, 0xb2}}	SA PEI GOP Init failure
{0x348cc7fe, 0x1e9a, 0x4c7a, {0x86, 0x28, 0xae, 0x48, 0x5b, 0x42, 0x10, 0xf0}}	SA PEI GOP GetMode failure
{0x5de1c071, 0x2c9c, 0x4a53, {0x80, 0x21, 0x4e, 0x80, 0xd2, 0x5d, 0x44, 0xa8}}	MRC training failure

When the FSP calls ReportStatusCode(), the Type parameter's EFI_STATUS_CODE_TYPE_MASK must be EFI_ERROR_CODE with the EFI_STATUS_CODE_SEVERITY_MASK >= EFI_ERROR_UNRECOVERED. The Value and Instance parameters must be 0.

The bootloader must register a listener for this status code. This listener should check if DataHeader.Type == STATUS_CODE_DATA_TYPE_FSP_ERROR_GUID to detect an FSP_ERROR_INFO notification. If an FSP_ERROR_INFO notification is encountered, the bootloader should assume that normal operation is no longer possible. In debug scenarios, this notification should be considered an ASSERT.

4.4 Dispatch Mode Integration

Dispatch Mode Integration Notes:

1. The FSP for this platform contains PEIMs compiled for the IA32 architecture. The boot loader therefore must utilize a PEI Foundation compiled for the IA32 architecture.
2. Since the FSP binary can be integrated into flash at any address, the boot loader has to report FSP FVs to the PEI and DXE dispatcher using PI specification defined mechanisms so PEIMs and DXE drivers inside the FSP Binary can be dispatched. FspmWrapperPeim and FspsWrapperPeim from IntelFsp2WrapperPkg can aid in implementing this.
3. For this platform, FSP-T, FSP-M, and FSP-S contain 1 FV each.
4. The FSP distribution package will include a DSC file which contains all DynamicEx PCDs consumed by the FSP binary. The boot loader should include the DSC during its build process so that any PCDs defined by this DSC file are included in the boot loader's PCD database. This enables the boot loader and FSP to share a single PCD database.
   - A NULL library (FspPcdListLibNull.inf) is included in the FSP distribution package. This library should be included in one of the boot loader's PEIMs. This ensures all DynamicEx PCDs used by the FSP are included in the boot loader's PCD database. One can fulfill this requirement by including the following code snippet in *BoardPkg.dsc:

```
IntelFsp2WrapperPkg/FspmWrapperPeim/FspmWrapperPeim.inf {
  <LibraryClasses>
  !if gIntelFsp2WrapperTokenSpaceGuid.PcdFspModeSelection == 0
    #
    # In FSP Dispatch mode below dummy library should be linked to bootloader PEIM
    # to build all DynamicEx PCDs that FSP consumes into bootloader PCD database.
    #
    NULL|$(PLATFORM_FSP_BIN_PACKAGE)/Library/FspPcdListLib/FspPcdListLibNull.inf
  !endif
}
```

5. The boot loader must provide at minimum 256KB of stack and 128KB of HOB heap to execute FSP on this platform.
6. In dispatch mode, the boot loader should not use FSP API calls described in chapter 5 of this document or chapter 8 of the FSP External Architecture Specification version 2.2. The TempRamInit API is the only exception, it is supported in both API mode and dispatch mode. All other APIs (MemoryInit, SiliconInit, etc.) should not be invoked.
7. For dispatch mode, FSP contains x64 DXE drivers to replace the NotifyPhase API. This eliminates thunking from 64bit to 32bit when using FSP dispatch mode. The boot loader should remove S3EndOfPeiNotify and FspWrapperNotifyDxe since they are no longer used in dispatch mode.
8. EFI_PEI_CORE_FV_LOCATION_PPI should be installed by the boot loader's SEC phase. EFI_PEI_CORE_FV_LOCATION_PPI.PeiCoreFvLocation should point to the first Firmware Volume (FV) in FSP-M so the PeiCore inside FSP will be invoked. If EFI_PEI_CORE_FV_LOCATION_PPI is not installed or PeiCore cannot be found at the address specified by EFI_PEI_CORE_FV_LOCATION_PPI.PeiCoreFvLocation, the PeiCore from the Boot Firmware Volume (BFV) will be invoked instead.

9. FSP-S requires multi-threaded code to complete silicon initialization on this platform. FSP-S includes the `UefiCpuPkg/CpuMpPei/CpuMpPei.inf` PEIM to provide multiprocessing. The bootloader can choose to either use the MP_SERVICES provided by the FSP for all PEIMs or the bootloader may provide an alternative implementation. In either case, only one MP_SERVICES implementation can be active at once. If the bootloader wishes to not use the FSP provided MP_SERVICES, then the bootloader must install the MP_SERVICES_PPI before installing the FV_INFO_PPI for FSP-S. If MP_SERVICES_PPI already exists, then FSP-S will use it and not execute its own implementation.

10. If you are using the Intel Reference BIOS, some EDK2 overrides may be required for Dispatch Mode, please refer to override folders in reference code or the override EDK2 repo for more details. For open source MinPlatform based firmware, no EDK2 overrides are required.

11. FSPM_ARCH_CONFIG_PPI->NvsBufferPtr is now a universal policy option (FSP Dispatch Mode and EDK2 native.) To enable the fast MRC training flow, the boot loader or platform code must to install this PPI to restore the previous MRC training data (SA_MISC_PEI_PREMEM_CONFIG->S3DataPtr is obsolete).

12. Policy initialization Flow Changes:

    - PEIMs from FSP-M/FSP-S should be dispatched early in PEI to produce the *DefaultPolicyInit* PPIs. -> Bootloader consumes the *DefaultPolicyInit* PPIs produced by the FSP binary to create the policy PPIs with default settings. -> Bootloader then locates and updates the policy PPIs as needed. -> Bootloader installs the *PolicyReadyPpi* after policy updates are completed. This signals to the FSP that silicon initialization may proceed.
    - Bootloader shall consume two PPIs produced by FSP binary to create policy PPIs with default settings. These PPIs are:
      - PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI
      - PEI_SI_DEFAULT_POLICY_INIT_PPI
    - The bootloader shall call the two functions below after the bootloader has completed any needed policy updates:
      - SiPreMemInstallPolicyReadyPpi()
      - SiInstallPolicyReadyPpi()

13. Debug message handling in dispatch mode:

    - Before the ReportStatusCode service is ready, a debug built FSP will send debug messages using the FSP-T UPD configuration (passed as FSP-T API input parameter). FSP-T is recommended to be used regardless FSP API mode or Dispatch mode.
    - Once the ReportStatusCode service is ready, a debug built FSP will send debug messages using the ReportStatusCode service.
    - It is recommended that bootloader register a StatusCode listener immediately after the ReportStatus↩ Code service is ready. It is important to register this listener as soon as possible so that all debug messages sent by the FSP are captured.
    - Please refer to section 9.4.7 in the Intel(R) Firmware Support Package External Architecture Specification v2.2 for details about the ReportStatusCode debug message format.

14. Enable or Disable FSP S3BootScript functionality: Since edk2-stable201911 release the PiDxeS3Boot↩ ScriptLib supports enabling or disabling S3BootScript functionality by PCD, and FSP consumes this PCD as DynamicEx type to enable or disable internal S3BootScript functionality. Bootloader must configure below PCD before entering DXE phase or before executing FSP DXE drivers.

    - gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiS3Enable = TRUE if S3BootScript should be enabled.
    - gEfiMdeModulePkgTokenSpaceGuid.PcdAcpiS3Enable = FALSE if S3BootScript should be disabled.

Chapter 5

FSP API Mode

Overview

This release of the FSP supports the all APIs required by the FSP External Architecture Specification version 2.2. These APIs are only used when running the FSP in API mode. In Dispatch mode, these APIs are not used (with the exception of TempRamInit.) The FSP information header contains the address offset for these APIs. Register usage is described in the FSP External Architecture Specification version 2.2. Any usage not described by the specification is described in the individual sections below.

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

5.1 FSP APIs

5.1.1 TempRamInit API

Please refer Chapter 8.6 in the FSP External Architecture Specification version 2.2 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, the last PcdFspReservedBufferSize bytes of space are reserved by the FSP for TempRamInit if temporary RAM initialization is done by the FSP. Any remaining space from **Temporary↩ RamBase**(ECX) to **TemporaryRamBase+TemporaryRamSize-FspReservedBufferSize** (EDX) is available for both bootloader and FSP use.

TempRamInit∗∗ also sets up the code caching of the region passed in through CodeCacheBase and CodeCache↩ Length, which are input parameters to TempRamInitApi. if 0 is passed in for CodeCacheBase, the base used will be (4 GB - 1 - CodeCacheLength).

Note

: When programming MTRRs CodeCacheLength will be reduced, if the LLC size on the current processor is smaller than the requested size.

It is a requirement for Firmware to have a Firmware Interface Table (FIT). The FIT contains pointers to each microcode update. The microcode update is loaded for all logical processors before executing the reset vector. If more than one 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 Temp↩ RamInit API. If these values are 0, FSP will not attempt to update microcode. If MicrocodeRegionBase != 0 and MicrocodeRegionLength != 0, then FSP will search that region for microcode updates. If a newer microcode update revision is found in the region, FSP will load it.

TempRamInit∗∗ will program MTRRs values to provide the following memory map:

Memory range	Cache Attribute
0xFEF00000 - 0x00040000	Write back
CodeCacheBase - CodeCacheLength	Write protect

5.1.2 FspMemoryInit API

Please refer to Chapter 8.7 in the FSP External Architecture Specification version 2.2 for the prototype, parameters and return value details for this API.

The variable **FspmUpdPtr** is a pointer to the **FSPM_UPD** structure which is described in the header file FspmUpd.h.

The bootloader must pass valid a CAR region for FSP through **FSPM_UPD.FspmArchUpd.StackBase** and **FSP↩M_UPD.FspmArchUpd.StackSize** UPDs.

The FSP for this platform will run FspMemoryInit top of the stack provided by the bootloader instead of establishing a separate stack as described by the FSP External Architecture Specification version 2.1/2.2. The memory region provided by the **FSPM_UPD.FspmArchUpd.StackBase** and **FSPM_UPD.FspmArchUpd.StackSize** UPDs is used to establish a HOB heap. The names **StackBase** and **StackSize** can be confusing since they are **NOT** used for stack. These names were retained for backwards compatibility with FSP v2.0.

Below are the heap and stack requirements for FSP on this platform:

HOB Heap requirement:

HOB Heap	UPD	Setting
Base	FSPM_UPD.FspmArchUpd.StackBase	Any non-conflict CAR region (0xFEF17F00 as default)
Size	FSPM_UPD.FspmArchUpd.StackSize	at least 128KB

Stack requirement: FSP's stack usage starts from the current stack pointer. The minimum stack size requirement for FSP-M is 256KB.

The bootloader must ensure that sufficient stack space is available to fulfill the FSP-M minimum stack size requirement at the point in execution where FspMemoryInit() is called. The stack allocated by the bootloader must be large enough for both FSP-M as well as any other parent function calls that are still on the stack at the point when FspMemoryInit() is called.

After FspMemoryInit() is completed, permanent memory is available. After this point, the memory pressure experienced early in boot is eliminated. Accordingly, right before FspMemoryInit() exits, any data that needs to be retained for later use by FspSiliconInit() will be copied to permanent memory. FspSiliconInit() will then execute on a second stack.

The base address of HECI device (Bus 0, Device 22, Function 0) is required to be initialized prior to calling Fsp↩MemoryInit(). The default address is programmed to 0xFED1A000.

FspMemoryInit() will calculate the memory map by taking into account the size of several memory regions: TSEG, IED, GTT, BDSM, ME stolen, Uncore PMRR, IOT, MOT, DPR, REMAP, TOLUD, TOUUD. These memory regions may not be initialized by FspMemoryInit(), but space will be reserved for them.

5.1.3 TempRamExit API

Please refer to Chapter 8.8 in the FSP external Architecture Specification version 2.2 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 bootloader 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.

The FSP for this platform doesn't have any input parameters for this API. The value of *TempRamExitParamPtr* should be NULL.

At the end of *TempRamExit* the original code and data cache are disabled. FSP will reconfigure all MTRRs as described in the table below. These MTRR values optimize performance in most scenarios. If the boot loader wishes to configure the MTRRs differently, they can be reprogrammed immediately after this API call.

Memory range	Cache Attribute
0xFF000000 - 0xFFFFFFFF (Flash region)	Write protect
0x00000000 - 0x0009FFFF	Write back
0x000C0000 - Top of Low Memory	Write back
xxxx - xxxx	x ∗Note1
0x100000000 - Top of High Memory	Write back ∗Note2

Stack requirement: 4KB of free stack space should be provided to execute **TempRamExit**.

Note1: Certain silicon features require specific cache types for specific memory ranges. These ranges will be configured by FSP when such features are enabled.

Note2: In some cases MTRRs might not be enough to cover all desired regions, in this case memory regions need to be adjusted for better alignment (e.g., adjust MmioSize or MmioSizeAdjustment UPD) Covering flash region and above 4GB memory is another case which may consume more MTRRs, when there is not enough MTRR available FSP will only cover above 4GB memory partially. In this case the boot loader can optimize MTRRs to remove flash from the cached regions after all needed data is loaded from flash and before booting the OS.

5.1.4 FspSiliconInit API

Please refer to Chapter 8.9 in the FSP external Architecture Specification version 2.2 for the prototype, parameters and return value details for this API.

The variable *FspsUpdPtr* is a pointer to the **FSPS_UPD** structure which is described in the header file FspsUpd.h.

It is expected that the boot loader will adjust MTRRs for SBSP if needed after **TempRamExit** but before entering **FspSiliconInit**. If the MTRRs are not programmed properly, boot performance can be impacted.

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

FspSiliconInit requires multi-threaded code to complete silicon initialization on this platform. FSP includes the `UefiCpuPkg/CpuMpPei/CpuMpPei.inf` PEIM to provide multiprocessing. Accordingly, the boot loader must expect FSP to perform an INIT-SIPI-SIPI during **FspSiliconInit** and **NotifyPhase** which will take control of all APs in the system and restart them. This will kill any background threads that were running on the APs. In some cases, this may not be desirable. As an alternative, the boot loader may provide an instance of the EFI_PEI_MP_SERVICES_PPI through the `FspsUpd->FspsConfig.CpuMpPpi` UPD for the FSP to use instead of the built-in implementation. **This is entirely optional**, if callbacks from FSP to the boot loader are not desired, one may set `FspsUpd->FspsConfig.CpuMpPpi == NULL`.

In summary, there are two methods that FSP can use for performing multiprocessing:

1. Using the multiprocessing services built-in to the FSP.

2. Using the boot loader's multiprocessing services.

Using method #1, the boot loader will set `FspsUpd->FspsConfig.CpuMpPpi == NULL`. If this UPD is NULL, then FSP will use its built-in MP services implementation for multiprocessing. Accordingly, the boot loader must expect FSP to perform an INIT-SIPI-SIPI during **FspSiliconInit** and **NotifyPhase** which will take control of all APs in the system and restart them. This will kill any background threads that were running on the APs.

Note: If bootloader needs to take control of APs back, a full AP re-initialization is required after FSP-S completed and control has been transferred back to bootloader.

Using method #2, the boot loader will set `FspsUpd->FspsConfig.CpuMpPpi != NULL`. If this UPD is not NULL, it must point to an instance of EFI_PEI_MP_SERVICES_PPI. When the FSP needs to perform multiprocessing, it will use the EFI_PEI_MP_SERVICES_PPI instance provided by the boot loader to do so. Accordingly, the boot loader must expect the function pointers in EFI_PEI_MP_SERVICES_PPI to be invoked in the middle of the execution of **FspSiliconInit** and **NotifyPhase**.

It is a requirement for the bootloader to have a 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, FSP will search that region for microcode updates. If a newer microcode update revision is found in the region, FSP will load it.

FspSiliconInit initializes PCH audio. This includes selecting the HD Audio verb table and initializing the audio CODEC.

FspSiliconInit initializes the following PCH features: HECI, USB, HSIO, Integrated Sensor Hub, Camera, PCI Express, DMI, Vt-d.

FspSiliconInit initializes the following CPU features: XD, VMX, AES, IED, HDC, x(2)APIC, Intel® Processor Trace, Three Strike Counter, Machine Check, Cache Pre-fetchers, Core PMRR, and Power Management.

FspSiliconInit initializes Internal Graphics. During this initialization, FSP publishes the EFI_PEI_GRAPHICS_INFO_HOB and the EFI_PEI_GRAPHICS_DEVICE_INFO_HOB. These HOBs will not be published during S3 resume as FSP will not initialize the internal graphics frame buffer during S3 resume.

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

Stack requirement: 4KB of free stack space should be provided to execute *FspSiliconInit*.

5.1.5 FspMultiPhaseSiInit API

Please refer Chapter 8.10 in the FSP External Architecture Specification version 2.2 for the prototype, parameters and return value details for this API.

If the FspMultiPhaseSiInit API is enabled (by setting FSPS_ARCH_UPD->EnableMultiPhaseSiliconInit UPD to 1 before the FspSiliconInit API is invoked), the bootloader should call this API with the MultiPhaseAction parameter set to 0 to get the NumberOfPhases supported by this platform. FSP will return control back to the bootloader NumberOfPhases times and then the bootloader should call FspMultiPhaseSiInit API with the MultiPhaseAction parameter set to 1 by NumberOfPhases times.

Note

: Disable FSPS_ARCH_UPD->EnableMultiPhaseSiliconInit when we disable FSP_M_CONFIG->TcssXhciEn.

This platform supports the following MultiPhaseSiInit phase(s):

Phase	FSP return point	Purpose
1	After TCSS initialization completed	for TCSS board specific initialization from bootloader side

5.1.6 NotifyPhase API

Please refer Chapter 8.11 in the FSP External Architecture Specification version 2.2 for the prototype, parameters and return value details for this API.

Stack requirement: 4KB of free stack space should be provided to execute *NotifyPhase*.

5.1.6.1 PostPciEnumeration Notification

This phase *EnumInitPhaseAfterPciEnumeration* is to be called after PCI enumeration but before execution of third party code such as option ROMs. It includes powering down unused PCH SATA ports, locking registers in P↩ MC, DMI, TCO, and SPI Flash (DLOCK + WRSDIS + FLOCKDN). It also includes enabling bus hosting for eSPI connected devices.

5.1.6.2 ReadyToBoot Notification

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

5.1.6.3 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.

5.1.7 FSP Events API

Please refer Chapter 8.5 in the FSP External Architecture Specification version 2.2 for the prototype, parameters and return value details for these APIs.

This platform supports FSP Events which re-direct debug messages to event handlers provided by bootloader. To enable this function the bootloader must provide event handler function pointers via UPDs.

FSP phase	UPD for bootloader to pass handler function pointer
FSP-T	FSPT_ARCH_UPD -> FspDebugHandler
FSP-M	FSPM_ARCH_UPD -> FspEventHandler
FSP-S	FSPS_ARCH_UPD -> FspEventHandler

5.2 Reset Return Codes

As per FSP External Architecture Specification version 2.0/2.1/2.2, any reset required in the FSP flow will be reported by returning one of the FSP_STATUS_RESET_REQUIRED∗ return codes.

It is the boot loader's responsibility to reset the system according to the reset type requested.

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

FSP_STATUS_RESET_REQUIRED Code	Reset Type requested
0x40000001	Cold Reset
0x40000002	Warm Reset
0x40000003	Global Reset - Puts the system through a Global Reset through HECI or a Full Reset through PCH
0x40000004	Reserved
0x40000005	Reserved
0x40000006	Reserved
0x40000007	Reserved
0x40000008	Reserved

5.3 UPD Porting Guide

Recommended values for UPDs:

UPD	Dependency	Description	Value
CstateLatencyControl1Irtl	Server platform	Server platform should has 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
BiosSize	Board design	Different board requires different value	tune

Chapter 6

Porting Recommendations

Here are some notes and recommendations for adapting an existing boot loader to FSP.

6.1 Locking PAM register

FSP 2.0 introduced the EndOfFirmware Notify phase callback which is the recommended place for locking PAM registers. Accordingly, by default FSP locks the PAM registers during the EndOfFirmware Notify phase. If the EndOf↩ Firmware Notify phase is too early to lock PAM registers, then the PAM locking code inside the FSP can be disabled by setting the UPD -> FSP_S_TEST_CONFIG -> SkipPamLock or SA policy -> _SI_PREMEM_POLICY_STRUCT -> SA_MISC_PEI_CONFIG -> SkipPamLock. If the PAM locking code inside the FSP is skipped, then the boot loader must lock the PAM registers before booting the OS. by programming one PCI config space register as below.

The PAM registers must be locked in all boot paths including S3 resume. To lock the PAM registers, program the following lock bit:

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

6.2 Locking SMRAM register

It is recommended that SMRAM be locked before any third party code (e.g. OpROM) execution. The point in execution where third party OpROMs begin executing can vary depending on the boot loader implementation. To provide flexibility, the FSP by default will not lock it. The boot loader should lock SMRAM by programming the following lock bit before any third party OpROM execution. Additionally, SMRAM should be locked before the **EnumInitPhaseReadyToBoot** notify phase is called. During S3 resume, the lock bit should be set right before the OS wake vector.

```
PciOr8 (B0: D0: F0: Register 0x88, BIT4);
Note: This register must be programmed using the legacy CF8/CFC PCI access mechanism. (MMIO access will not
       work)
```

6.3 Locking SMI register

It is recommended that the global SMI bit is locked before any third party code (e.g. OpROM) execution. SMM initialization flows may vary depending on boot loader implementation details. Accordingly, FSP will not lock it by default. The boot loader is responsible for locking the following registers after SMM configuration is complete. Set AcpiBase + 0x30[0] to 1b to enable global SMI. Set PMC PCI offset A0h[4] = 1b to lock SMI.

6.4 Verify below settings are correct for your platforms

PMC PCI configuration space is not PCI specification compliant. The 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 these addresses in the boot loader code instead of reading BAR addresses from the PMC controller.

Register	Values
ABASE	0x1800
PWRMBASE	0xFE000000
PCIEXBAR_BASE_ADDRESS	0xE0000000

Note

:

- Boot Loader can use a different value for PCIEXBAR_BASE_ADDRESS either by modifying the UPD (under FSP-T) or by overriding the PCIEXBAR (B0:D0:F0:R60h) before calling FspMemoryInit.
- Boot Loader should avoid using conflicting address when reprogramming PCIEXBAR_BASE_ADDRESS than the recommended one.

Chapter 7

FSP Output

The FSP builds a series of data structures called 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 Chapter 10 in the FSP External Architecture Specification version 2.2 for details about FSP Architectural HOBs.

The sections below describe the HOBs not covered in the above two specifications.

7.1 Resource Descriptor HOB

Below resource HOBs with specific owner GUID are publicized to specify memory ranges used by FSP. Bootloader shall check and not consume these ranges to avoid memory collision.

7.1.1 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 } }
```

7.1.2 TraceHub Resource Descriptor HOB

TraceHub Resource HOB is reported when TraceHub is enabled or PlatformDebugConsent UPD is 2 (Enabled) to reserve memory above 4G for TraceHub use.

```
#define FSP_HOB_RESOURCE_OWNER_TRACEHUB_GUID  \
{ 0x5fb35909, 0x5a1c, 0x4a31, { 0xad, 0xaf, 0x57, 0x7b, 0x54, 0x68, 0x26, 0x3f } }
```

7.2 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, & 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 \
  { 0x6210d62f, 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 \
  { 0x798e722e, 0x15b2, 0x4e13, { 0x8a, 0xe9, 0x6b, 0xa3, 0x0f, 0xf7, 0xf1, 0x67 }}

///
/// 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;

///
/// The Smbios structure header.
///
typedef struct {
  UINT8                         Type;
  UINT8                         Length;
  UINT16                        Handle;
} SMBIOS_STRUCTURE;
///
/// Firmware Version Information HOB Structure
///
typedef struct {
  SMBIOS_STRUCTURE            Header;         ///< SMBIOS structure header
  UINT8                       Count;          ///< Number of FVI entries in this structure
```

```
  INTEL_FIRMWARE_VERSION_INFO Fvi[1];         ///< FVI structure(s)
///
/// FIRMWARE_VERSION_INFO structures followed by the null terminated string buffer
///
} FIRMWARE_VERSION_INFO_HOB;
```

7.3 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 FspsUpd.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;
```

7.4 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 much temporary RAM is 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;
```

7.5 FSP_ERROR_INFO_HOB

In the case of an error occurring during the execution of the FSP, the FSP may produce this HOB which describes the error in more detail. FSP_ERROR_INFO_HOB is only used in FSP API Mode. In FSP Dispatch Mode, FSP may call ReportStatusCode() and provide a FSP_ERROR_INFO structure using the PI status code services.

```
#define FSP_ERROR_INFO_HOB_GUID \
{0x611e6a88, 0xadb7, 0x4301, { 0x93, 0xff, 0xe4, 0x73, 0x04, 0xb4, 0x3d, 0xa6 }}
typedef struct {
  EFI_HOB_GUID_TYPE     GuidHob;
  EFI_STATUS_CODE_TYPE  Type;
  EFI_STATUS_CODE_VALUE Value;
  UINT32                Instance;
  EFI_GUID              CallerId;
  EFI_GUID              ErrorType;
  UINT32                Status;
} FSP_ERROR_INFO_HOB;
```

The FSP for this platform implements the following CallerId GUIDs:

CallerId	Description
{0x1f4dc7e9, 0x26ca, 0x4336, {0x8c, 0xe3, 0x39, 0x31, 0x03, 0xb5, 0xf3, 0xd7}}	ME
{0x98230916, 0xe632, 0x49ff, {0x81, 0x81, 0x55, 0xce, 0xe5, 0x10, 0x36, 0x89}}	System Agent
{0x5a47c211, 0x642f, 0x4f92, {0x9c, 0xb3, 0x7f, 0xeb, 0x93, 0xda, 0xdd, 0xba}}	MRC

The following ErrorType GUIDs are implemented:

ErrorType	Description
{0x948585c4, 0x76a4, 0x45bb, {0xbe, 0x6c, 0x39, 0x61, 0xc3, 0xab, 0xde, 0x15}}	ME EOP failure
{0x8106a5cc, 0x30ba, 0x41cf, {0xa1, 0x78, 0x63, 0x38, 0x91, 0x11, 0xae, 0xb2}}	SA PEI GOP Init failure
{0x348cc7fe, 0x1e9a, 0x4c7a, {0x86, 0x28, 0xae, 0x48, 0x5b, 0x42, 0x10, 0xf0}}	SA PEI GOP GetMode failure
{0x5de1c071, 0x2c9c, 0x4a53, {0x80, 0x21, 0x4e, 0x80, 0xd2, 0x5d, 0x44, 0xa8}}	MRC training failure

Chapter 8

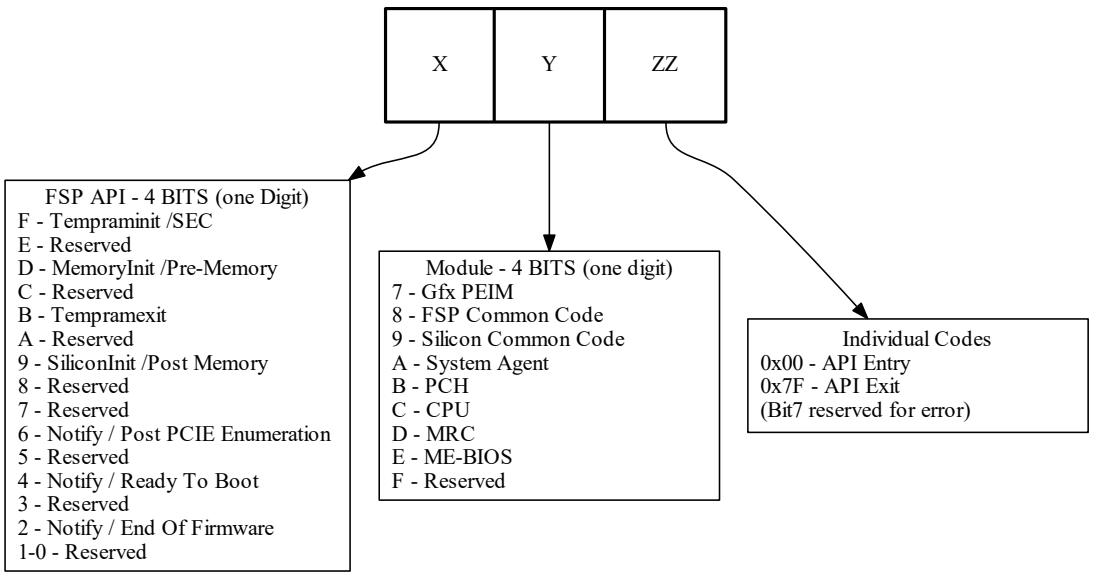
POST Codes

The FSP outputs 16-bit POST codes to indicate which API and which module is currently executing.

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

8.1 POST Code Info

The diagram below illustrates how sub-fields are encoded in the POST codes produced by the FSP.



8.1.1 TempRamInit API Status Codes (0xFxxx)

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

8.1.2 FspMemoryInit API Status Codes (0xDxxx)

PostCode	Module	Description
0xD800	FSP	FspMemoryInit API Entry
0xD87F	FSP	FSpMemoryInit API Exit
0xDA00	SA	Pre-Mem SaInit 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	SaSvInit 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 SaInit Exit
0xDB00	PCH	Pre-Mem PchInit 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 PchInit Exit
0xDC00	CPU	CPU Pre-Mem Entry
0xDC0F	CPU	CpuAddPreMemConfigBlocks Done
0xDC20	CPU	CpuOnPolicyInstalled Start
0xDC2F	CPU	XmmInit Start
0xDC3F	CPU	TxtInit 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
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

PostCode	Module	Description
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
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

PostCode	Module	Description
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
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

8.1.3 TempRamExit API Status Codes (0xBxxx)

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

8.1.4 FspSiliconInit API Status Codes (0x9xxx)

PostCode	Module	Description
0x9800	FSP	FspSiliconInit API Entry
0x987F	FSP	FspSiliconInit API Exit
0x9A00	SA	PostMem SaInit Entry
0x9A01	SA	DeviceConfigure Start
0x9A02	SA	UpdateSaHobPostMem Start
0x9A03	SA	Initializing Pei Display
0x9A04	SA	PeiGraphicsNotifyCallback Entry
0x9A05	SA	CallPpiAndFillFrameBuffer
0x9A06	SA	GraphicsPpiInit
0x9A07	SA	GraphicsPpiGetMode
0x9A08	SA	FillFrameBufferAndShowLogo
0x9A0F	SA	PeiGraphicsNotifyCallback Exit
0x9A14	SA	Initializing SA IPU device
0x9A16	SA	Initializing SA GNA device
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 SaInit Exit
0x9A50	SA	SaSecurityLock Start
0x9A5F	SA	SaSecurityLock End
0x9A60	SA	SaSResetComplete Entry
0x9A61	SA	Set BIOS_RESET_CPL to indicate all configurations complete
0x9A62	SA	SaSvInit2 Start
0x9A63	SA	GraphicsPmInit Start
0x9A64	SA	SaPciPrint Start
0x9A6F	SA	SaSResetComplete Exit
0x9A70	SA	SaS3ResumeAtEndOfPei Callback Entry
0x9A7F	SA	SaS3ResumeAtEndOfPei Callback Exit
0x9B00	PCH	Post-Mem PchInit 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
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

PostCode	Module	Description
0x9B14	PCH	Post-Mem ISH initization
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	CpuInit Start
0x9C17	CPU	CollectProcessorFeature Start
0x9C18	CPU	ProgramProcessorFeature Start
0x9C19	CPU	ProgramProcessorFeature Done
0x9C20	CPU	CpuInitPreResetCpl Start
0x9C21	CPU	ProcessorsPrefetcherInitialization Start
0x9C22	CPU	InitRatl Start
0x9C23	CPU	ConfigureSvidVrs Start
0x9C24	CPU	ConfigurePidSettings Start
0x9C25	CPU	SetBootFrequency Start
0x9C26	CPU	CpuOcInitPreMem Start
0x9C27	CPU	CpuOcInit Reset
0x9C28	CPU	BiosGuardInit Start
0x9C29	CPU	BiosGuardInit Reset
0x9C3F	CPU	CpuInitPreResetCpl Done
0x9C42	CPU	ApSafePostMicrocodePatchInit Start
0x9C43	CPU	InitializeCpuDataHob Start
0x9C44	CPU	InitializeCpuDataHob Done
0x9C4F	CPU	CpuInit 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

8.1.5 NotifyPhase API Status Codes (0x6xxx)

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

Chapter 9

Todo List

Page Config Blocks

This list is from TigerLake, needs to be updated for MeteorLake

This list is from TigerLake, needs to be updated for MeteorLake

Chapter 10

Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 11

File Index

11.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 12

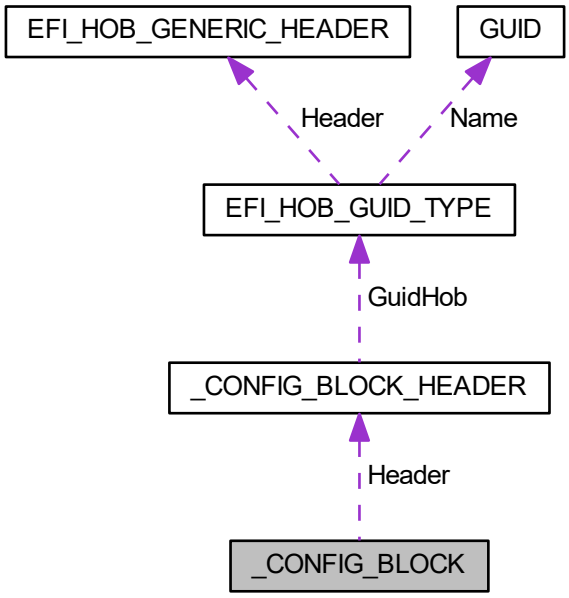
Class Documentation

12.1 _CONFIG_BLOCK Struct Reference

Config Block.

`#include <ConfigBlock.h>`

Collaboration diagram for _CONFIG_BLOCK:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Header of config block.

12.1.1 Detailed Description

Config Block.

Definition at line 40 of file ConfigBlock.h.

The documentation for this struct was generated from the following file:

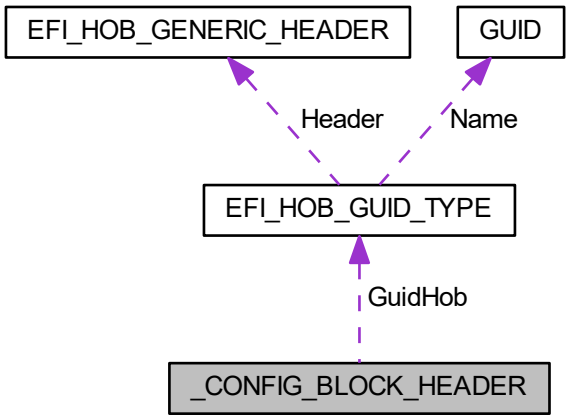
- ConfigBlock.h

12.2 _CONFIG_BLOCK_HEADER Struct Reference

Config Block Header.

```
#include <ConfigBlock.h>
```

Collaboration diagram for _CONFIG_BLOCK_HEADER:



Public Attributes

- EFI_HOB_GUID_TYPE GuidHob

  Offset 0-23 GUID extension HOB header.
- UINT8 Revision

  Offset 24 Revision of this config block.
- UINT8 Attributes

  Offset 25 The main revision for config block.
- UINT8 Reserved [2]

  Offset 26-27 Reserved for future use.

12.2.1 Detailed Description

Config Block Header.

Definition at line 30 of file ConfigBlock.h.

The documentation for this struct was generated from the following file:

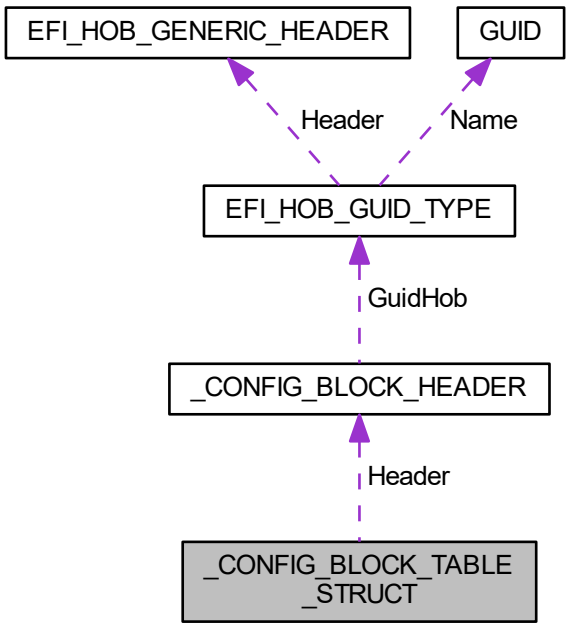
- ConfigBlock.h

12.3 _CONFIG_BLOCK_TABLE_STRUCT Struct Reference

Config Block Table Header.

`#include <ConfigBlock.h>`

Collaboration diagram for _CONFIG_BLOCK_TABLE_STRUCT:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 GUID number for main entry of config block.
- UINT8 Rsvd0 [2]

  Offset 28-29 Reserved for future use.
- UINT16 NumberOfBlocks

  Offset 30-31 Number of config blocks (N)
- UINT32 AvailableSize

  Offset 32-35 Current config block table size.

12.3.1 Detailed Description

Config Block Table Header.

Definition at line 50 of file ConfigBlock.h.

The documentation for this struct was generated from the following file:

- ConfigBlock.h

12.4 _EFI_MM_RESERVED_MMRAM_REGION Struct Reference

Structure describing a MMRAM region which cannot be used for the MMRAM heap.

`#include <PiMultiPhase.h>`

Public Attributes

- EFI_PHYSICAL_ADDRESS MmramReservedStart
  Starting address of the reserved MMRAM area, as it appears while MMRAM is open.
- UINT64 MmramReservedSize
  Number of bytes occupied by the reserved MMRAM area.

12.4.1 Detailed Description

Structure describing a MMRAM region which cannot be used for the MMRAM heap.

Definition at line 139 of file PiMultiPhase.h.

12.4.2 Member Data Documentation

12.4.2.1 MmramReservedSize

`UINT64 _EFI_MM_RESERVED_MMRAM_REGION::MmramReservedSize`

Number of bytes occupied by the reserved MMRAM area.

A size of zero indicates the last MMRAM area.

Definition at line 149 of file PiMultiPhase.h.

12.4.2.2 MmramReservedStart

```
EFI_PHYSICAL_ADDRESS _EFI_MM_RESERVED_MMRAM_REGION::MmramReservedStart
```

Starting address of the reserved MMRAM area, as it appears while MMRAM is open.

Ignored if MmramReservedSize is 0.

Definition at line 144 of file PiMultiPhase.h.

The documentation for this struct was generated from the following file:

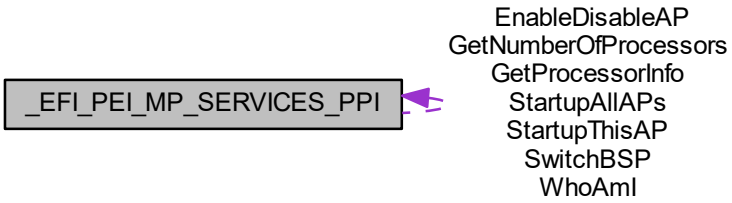
- PiMultiPhase.h

12.5 _EFI_PEI_MP_SERVICES_PPI Struct Reference

This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

```
#include <MpServices.h>
```

Collaboration diagram for _EFI_PEI_MP_SERVICES_PPI:



12.5.1 Detailed Description

This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

Definition at line 265 of file MpServices.h.

The documentation for this struct was generated from the following file:

- MpServices.h

12.6 _ITBT_GENERIC_CONFIG Struct Reference

ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

```
#include <PeiITbtGenericStructure.h>
```

Public Attributes

- UINT16 ITbtForcePowerOnTimeoutInMs

  Timeout value for forcing power iTBT controller on every boot/reboot/Sx exit as a precondition for execution of following mailbox communication.
- UINT16 ITbtConnectTopologyTimeoutInMs

  Timeout value while sending connect topology mailbox command in order to bring all connected TBT devices are available on PCIe before BIOS will enumerate them in BDS ***(Test) default is 5000 ms***
- UINT8 Reserved2

  Reserved and previously used for iTbt Security Level.
- UINT8 ITbtPcieTunnelingForUsb4

  Disable/Enable PCIe tunneling for USB4. ***default is enable***
- UINT8 Usb4CmMode

  USB4 CM mode.
- UINT8 Reserved [1]

  Reserved for DWORD alignment.

12.6.1 Detailed Description

ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

Definition at line 52 of file PeiITbtGenericStructure.h.

12.6.2 Member Data Documentation

12.6.2.1 ITbtForcePowerOnTimeoutInMs

```
UINT16 _ITBT_GENERIC_CONFIG::ITbtForcePowerOnTimeoutInMs
```

Timeout value for forcing power iTBT controller on every boot/reboot/Sx exit as a precondition for execution of following mailbox communication.

After applying Force Power Thunderbolt BIOS shall poll for iTBT readiness for mailbox communication If TBT cable is disconnected, iTBT microcontrollers are in lower power state. To ensure successful mailbox execution, independently on presence of TBT cable, TBT BIOS shall bring iTBT microcontrollers up by applying Force Power. iTBT microcontrollers will wake up either due to TBT cable presence or Force Power event. **(Test) default is 500 ms**

Definition at line 64 of file PeiITbtGenericStructure.h.

The documentation for this struct was generated from the following file:

- PeiITbtGenericStructure.h

12.7 _ITBT_ROOTPORT_CONFIG Struct Reference

iTBT RootPort Data Structure

```
#include <PeiITbtGenericStructure.h>
```

Public Attributes

- UINT8 ITbtPcieRootPortEn

  Disable/Enable iTBT PCIe Root Port.
- UINT8 Reserved [3]

  Reserved for DWORD alignment.

12.7.1 Detailed Description

iTBT RootPort Data Structure

Definition at line 44 of file PeiITbtGenericStructure.h.

The documentation for this struct was generated from the following file:

- PeiITbtGenericStructure.h

12.8 _LIST_ENTRY Struct Reference

_LIST_ENTRY structure definition.

```
#include <Base.h>
```

Collaboration diagram for _LIST_ENTRY:

12.8.1 Detailed Description

_LIST_ENTRY structure definition.

Definition at line 247 of file Base.h.

The documentation for this struct was generated from the following file:

- Base.h

12.9 _PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI Struct Reference

This PPI provides function to install default silicon policy.

`#include <PeiPreMemSiDefaultPolicy.h>`

Public Attributes

- PEI_PREMEM_POLICY_INIT PeiPreMemPolicyInit
  PeiPreMemPolicyInit()

12.9.1 Detailed Description

This PPI provides function to install default silicon policy.

Definition at line 55 of file PeiPreMemSiDefaultPolicy.h.

The documentation for this struct was generated from the following file:

- PeiPreMemSiDefaultPolicy.h

12.10 _PEI_SI_DEFAULT_POLICY_INIT_PPI Struct Reference

This PPI provides function to install default silicon policy.

`#include <PeiSiDefaultPolicy.h>`

Public Attributes

- PEI_POLICY_INIT PeiPolicyInit
  PeiPolicyInit()

12.10.1 Detailed Description

This PPI provides function to install default silicon policy.

Definition at line 55 of file PeiSiDefaultPolicy.h.

The documentation for this struct was generated from the following file:

- PeiSiDefaultPolicy.h

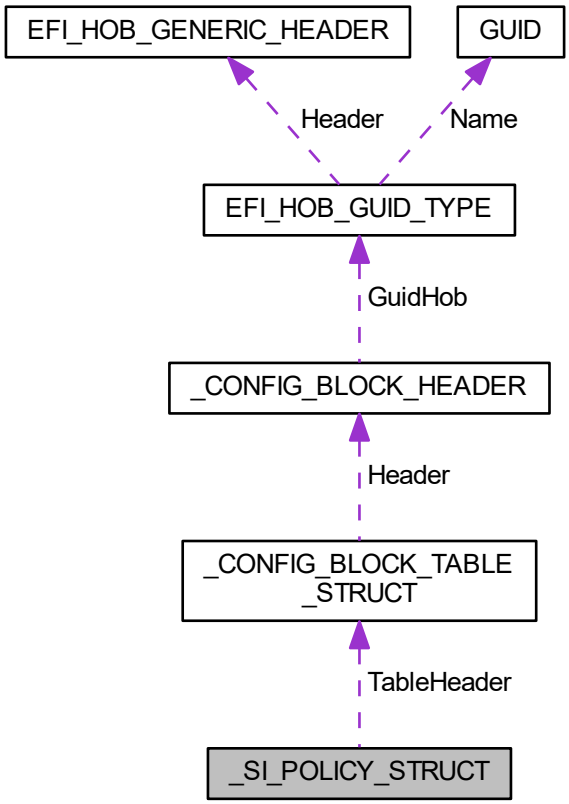
12.11 _SI_POLICY_STRUCT Struct Reference

SI Policy PPI
All SI config block change history will be listed here

.

```
#include <SiPolicyStruct.h>
```

Collaboration diagram for _SI_POLICY_STRUCT:



Public Attributes

- CONFIG_BLOCK_TABLE_HEADER TableHeader
  Config Block Table Header.

12.11.1 Detailed Description

SI Policy PPI
All SI config block change history will be listed here

.

- **Revision 1**:
  - Initial version.

Definition at line 85 of file SiPolicyStruct.h.

The documentation for this struct was generated from the following file:

- SiPolicyStruct.h

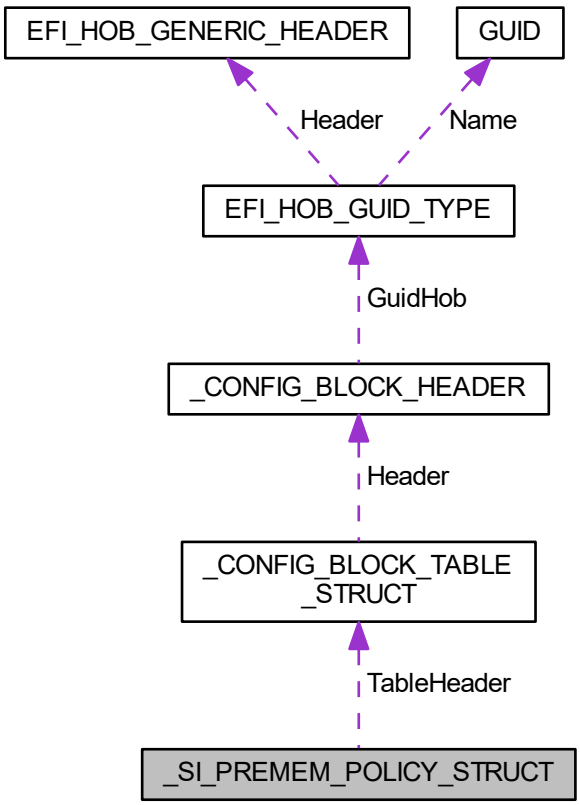
12.12 _SI_PREMEM_POLICY_STRUCT Struct Reference

SI Policy PPI in Pre-Mem
All SI config block change history will be listed here

.

```
#include <SiPolicyStruct.h>
```

Collaboration diagram for _SI_PREMEM_POLICY_STRUCT:



Public Attributes

- CONFIG_BLOCK_TABLE_HEADER TableHeader

  Config Block Table Header.

12.12.1 Detailed Description

SI Policy PPI in Pre-Mem
All SI config block change history will be listed here

.

- **Revision 1**:
  - Initial version.

Definition at line 71 of file SiPolicyStruct.h.

The documentation for this struct was generated from the following file:

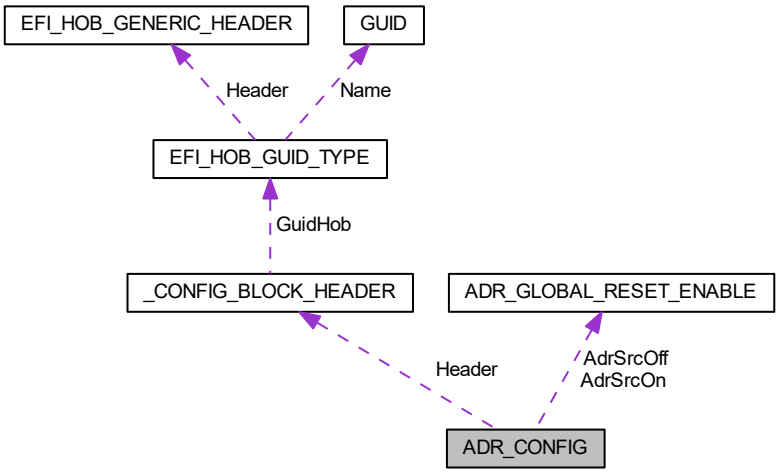
- SiPolicyStruct.h

12.13 ADR_CONFIG Struct Reference

ADR Configuration **Revision 1**: - Initial version.

```
#include <AdrConfig.h>
```

Collaboration diagram for ADR_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 AdrEn: 2

  Determine if Adr is enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.
- UINT32 AdrTimerEn: 2

  Determine if Adr timer options are enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.
- UINT32 AdrTimer1Val: 8

  Determines the Timeout value used for the ADR timer 1. A value of zero bypasses the timer.
- UINT32 AdrMultiplier1Val: 3

  Specifies the tick frequency upon which the timer 1 will increment. ADR_TIMER_SCALE should be used to encode values.
- UINT32 AdrTimer2Val: 8

  Determines the Timeout value used for the ADR timer 2. A value of zero bypasses the timer.
- UINT32 AdrMultiplier2Val: 3

  Specifies the tick frequency upon which the timer 2 will increment. ADR_TIMER_SCALE should be used to encode values.
- UINT32 AdrHostPartitionReset: 2

  Determine if Host Partition Reset is enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.
- UINT32 AdrPlatAckEn: 2

  Determine if Platform Acknowledge is enabled - 0: PLATFORM_POR, 1: FORCE_ENABLE, 2: FORCE_DISABLE.
- ADR_GLOBAL_RESET_ENABLE AdrSrcOff

  Determine which ADR sources are disabled. The code perform clearing bits in ADR Global Reset Enable register. This is first step.
- ADR_GLOBAL_RESET_ENABLE AdrSrcOn

  Determine which ADR sources are enabled. The code perform setting bits in ADR Global Reset Enable register. This is second step.

12.13.1 Detailed Description

ADR Configuration **Revision 1**: - Initial version.

Definition at line 97 of file AdrConfig.h.

The documentation for this struct was generated from the following file:

- AdrConfig.h

12.14 ADR_GLOBAL_RESET_ENABLE Union Reference

ADR Source Enable.

`#include <AdrConfig.h>`

12.14.1 Detailed Description

ADR Source Enable.

Definition at line 59 of file AdrConfig.h.

The documentation for this union was generated from the following file:

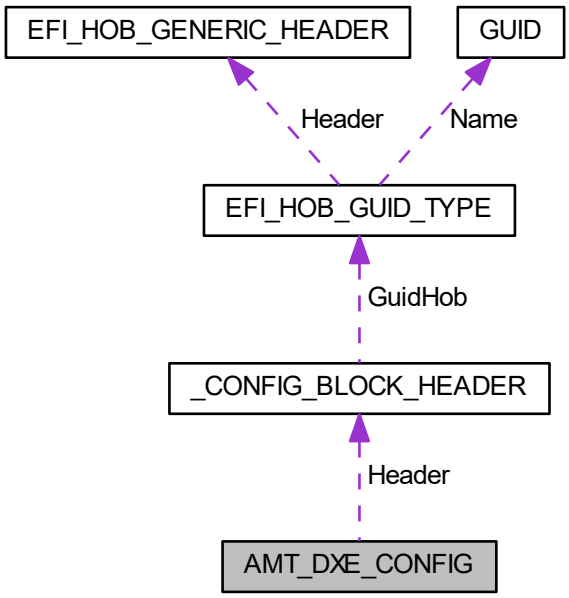
- AdrConfig.h

12.15 AMT_DXE_CONFIG Struct Reference

AMT Dxe Configuration Structure.

`#include <AmtConfig.h>`

Collaboration diagram for AMT_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 CiraRequest: 1
  *Trigger CIRA boot. **0: No CIRA request**; 1: Trigger CIRA request.*
- UINT32 UnConfigureMe: 1

OEMFlag Bit 15: Unconfigure ME with resetting MEBx password to default. ***0: No****; 1: Un-configure ME without password.*

- UINT32 UsbProvision: 1

  Enable/Disable of AMT USB Provisioning. ***0: Disable****; 1: Enable.*
- UINT32 RsvdBits: 29

  Reserved for future use & Config block alignment.
- AMT_REPORT_ERROR AmtReportError

  Function pointer for displaying error message on screen.

12.15.1 Detailed Description

AMT Dxe Configuration Structure.

Revision 1:

- Initial version.

Definition at line 130 of file AmtConfig.h.

The documentation for this struct was generated from the following file:

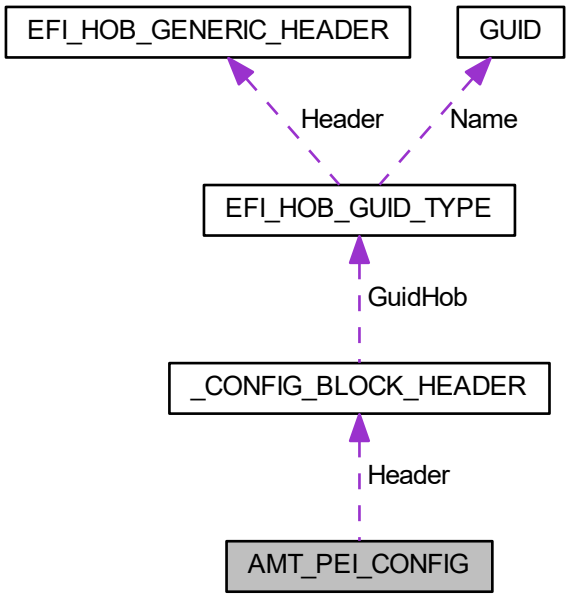
- AmtConfig.h

12.16 AMT_PEI_CONFIG Struct Reference

AMT Pei Configuration Structure.

`#include <AmtConfig.h>`

Collaboration diagram for AMT_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 AmtEnabled: 1

  Enable or Disable Intel Active Management Technology feature.
- UINT32 WatchDogEnabled: 1

  ME WatchDog timer feature.
- UINT32 FwProgress: 1

  PET Events Progress to receive PET Events. 0: Disable; ***1: Enable***
- UINT32 AmtSolEnabled: 1

  Serial Over Lan retrieved from Mebx. The default value depends on CSME/AMT. 0: Disable, ***1: Enable***
- UINT32 RsvdBits: 28

  Reserved for future use & Config block alignment.
- UINT16 WatchDogTimerOs

  OS WatchDog Timer ***0: Disable*** *OS WDT won't be started after stopping BIOS WDT even if WatchDogEnabled is 1.*
- UINT16 WatchDogTimerBios

  BIOS WatchDog Timer ***0: Disable*** *BIOS WDT won't be started even if WatchDogEnabled is 1.*

12.16.1 Detailed Description

AMT Pei Configuration Structure.

Revision 1:

- Initial version.

Definition at line 89 of file AmtConfig.h.

12.16.2 Member Data Documentation

12.16.2.1 AmtEnabled

`UINT32 AMT_PEI_CONFIG::AmtEnabled`

Enable or Disable Intel Active Management Technology feature.

If disabled, all Intel AMT features, including Alert Standard Format features, will not be supported. 0: Disable **1: Enable.**

Definition at line 97 of file AmtConfig.h.

12.16.2.2 WatchDogEnabled

```
UINT32 AMT_PEI_CONFIG::WatchDogEnabled
```

ME WatchDog timer feature.

If disabled, below WatchDogTimerOs/WatchDogTimerBios will be irrelevant. See WatchDogTimerOs and Watch↩ DogTimerBios description. **0: Disable** 1: Enable ME WDT if corresponding timer value is not zero.

Definition at line 104 of file AmtConfig.h.

12.16.2.3 WatchDogTimerBios

```
UINT16 AMT_PEI_CONFIG::WatchDogTimerBios
```

BIOS WatchDog Timer **0: Disable** BIOS WDT won't be started even if WatchDogEnabled is 1.

Non zero value - The BIOS WDT is set according to the value and started if WatchDogEnabled is 1.

Definition at line 120 of file AmtConfig.h.

12.16.2.4 WatchDogTimerOs

```
UINT16 AMT_PEI_CONFIG::WatchDogTimerOs
```

OS WatchDog Timer **0: Disable** OS WDT won't be started after stopping BIOS WDT even if WatchDogEnabled is 1.

Non zero value - OS WDT will be started after stopping BIOS WDT if WatchDogEnabled is 1. The timer is set according to the value.

Definition at line 114 of file AmtConfig.h.

The documentation for this struct was generated from the following file:

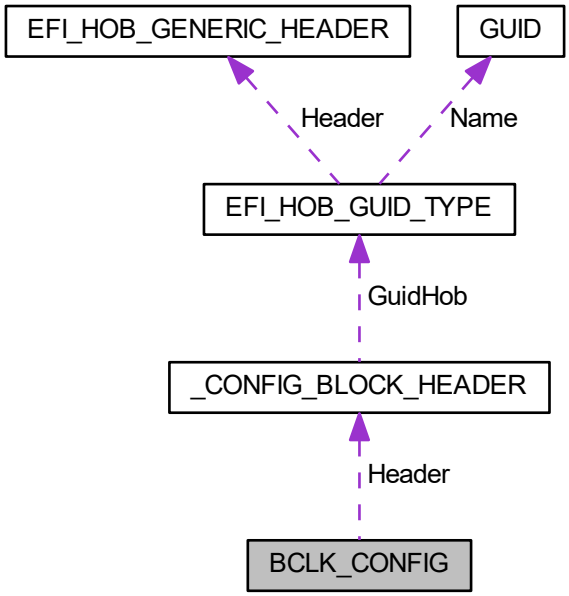
- AmtConfig.h

12.17 BCLK_CONFIG Struct Reference

BCLK configuration.

`#include <PmConfig.h>`

Collaboration diagram for BCLK_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 SocBclkPllOn

  Controls enable status of SOC BCLK.
- UINT8 CpuBclkPllOn

  Controls enable status of CPU BCLK.

12.17.1 Detailed Description

BCLK configuration.

Only available on projects with SOC and CPU BCLK supported.

Definition at line 497 of file PmConfig.h.

12.17.2 Member Data Documentation

12.17.2.1 CpuBclkPllOn

`UINT8 BCLK_CONFIG::CpuBclkPllOn`

Controls enable status of CPU BCLK.

0: HW default, 1: Force enable, 2: Force disable

Definition at line 509 of file PmConfig.h.

12.17.2.2 SocBclkPllOn

`UINT8 BCLK_CONFIG::SocBclkPllOn`

Controls enable status of SOC BCLK.

0: HW default, 1: Force enable, 2: Force disable

Definition at line 504 of file PmConfig.h.

The documentation for this struct was generated from the following file:

- PmConfig.h

12.18 CNVI_PIN_MUX Struct Reference

CNVi signals pin muxing settings.

`#include <CnviConfig.h>`

Public Attributes

- UINT32 RfReset

  RF_RESET# Pin mux configuration. Refer to GPIO_*_MUXING_CNVI_RF_RESET_*.
- UINT32 Clkreq

  CLKREQ Pin mux configuration. Refer to GPIO_*_MUXING_CNVI_*_CLKREQ_*.

12.18.1 Detailed Description

CNVi signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_C↩ NVI_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 81 of file CnviConfig.h.

The documentation for this struct was generated from the following file:

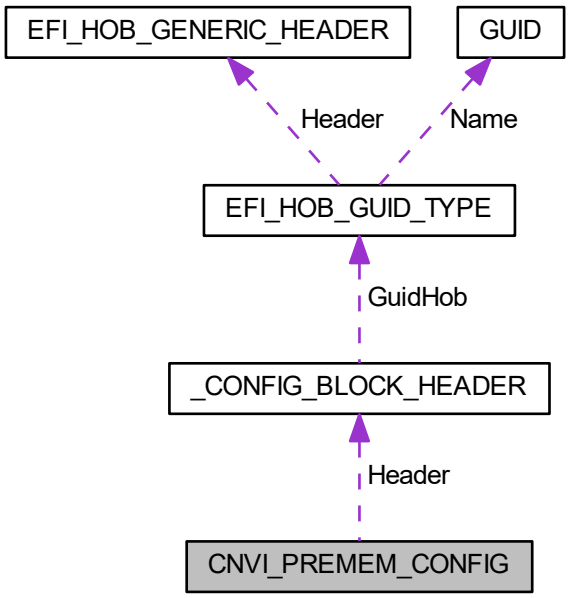
- CnviConfig.h

12.19 CNVI_PREMEM_CONFIG Struct Reference

The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP.

`#include <CnviConfig.h>`

Collaboration diagram for CNVI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 DdrRfim

  The option to enable or disable DDR RFI Mitigation. 0: Disabled, ***1: Enabled***

12.19.1 Detailed Description

The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP.

Revision 1: - Initial version.

Definition at line 126 of file CnviConfig.h.

The documentation for this struct was generated from the following file:

- CnviConfig.h

12.20 DMI_HW_WIDTH_CONTROL Struct Reference

This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design.

`#include <ThermalConfig.h>`

Public Attributes

- UINT32 DmiTsawEn: 1
  DMI Thermal Sensor Autonomous Width Enable.
- UINT32 SuggestedSetting: 1
  *0: Disable; **1: Enable** suggested representative values*
- UINT32 RsvdBits0: 6
  Reserved bits.
- UINT32 TS0TW: 3
  *Thermal Sensor 0 Target Width (**DmiThermSensWidthx8**)*
- UINT32 TS1TW: 3
  *Thermal Sensor 1 Target Width (**DmiThermSensWidthx4**)*
- UINT32 TS2TW: 3
  *Thermal Sensor 2 Target Width (**DmiThermSensWidthx2**)*
- UINT32 TS3TW: 3
  *Thermal Sensor 3 Target Width (**DmiThermSensWidthx1**)*
- UINT32 RsvdBits1: 12
  Reserved bits.

12.20.1 Detailed Description

This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design.

When the SuggestedSetting is enabled, the customized values are ignored. Look at DMI_THERMAL_SENSOR_↩ TARGET_WIDTH for possible values

Definition at line 83 of file ThermalConfig.h.

The documentation for this struct was generated from the following file:

- ThermalConfig.h

12.21 DMI_LANE_CONFIG Struct Reference

The PCH_DMI_LANE_CONFIG block describes specific lane configuration.

`#include <PchPcieRpConfig.h>`

Public Attributes

- UINT32 DmiTranCoOverEn: 2
  Enable/Disable Lane Transmitter Coefficient.
- UINT32 Rsvd1: 6
  Reserved.
- UINT32 DmiTranCoOverPostCur: 6
  Lane Transmitter Post-Cursor Coefficient Override.
- UINT32 Rsvd2: 2
  Reserved.
- UINT32 DmiTranCoOverPreCur: 6
  Lane Transmitter Pre-Cursor Coefficient Override.
- UINT32 Rsvd3: 2
  Reserved.
- UINT32 DmiUpPortTranPreset: 4
  Upstream Port Lane Transmitter Preset.
- UINT32 Rsvd4: 4
  Reserved.

12.21.1 Detailed Description

The PCH_DMI_LANE_CONFIG block describes specific lane configuration.

Definition at line 169 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

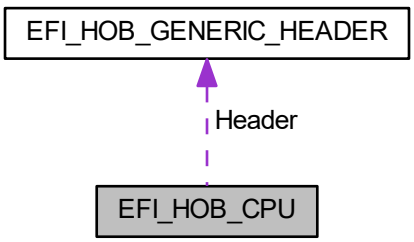
- PchPcieRpConfig.h

12.22 EFI_HOB_CPU Struct Reference

Describes processor information, such as address space and I/O space capabilities.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_CPU:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- UINT8 SizeOfMemorySpace

  Identifies the maximum physical memory addressability of the processor.
- UINT8 SizeOfIoSpace

  Identifies the maximum physical I/O addressability of the processor.
- UINT8 Reserved [6]

  This field will always be set to zero.

12.22.1 Detailed Description

Describes processor information, such as address space and I/O space capabilities.

Definition at line 445 of file PiHob.h.

12.22.2 Member Data Documentation

12.22.2.1 Header

`EFI_HOB_GENERIC_HEADER EFI_HOB_CPU::Header`

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_CPU.

Definition at line 449 of file PiHob.h.

The documentation for this struct was generated from the following file:

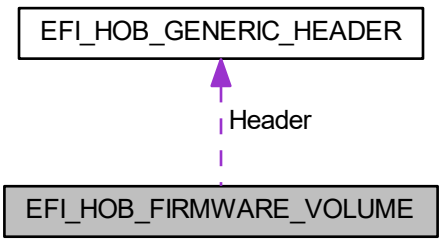
- PiHob.h

12.23 EFI_HOB_FIRMWARE_VOLUME Struct Reference

Details the location of firmware volumes that contain firmware files.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_FIRMWARE_VOLUME:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_PHYSICAL_ADDRESS BaseAddress

  The physical memory-mapped base address of the firmware volume.
- UINT64 Length

  The length in bytes of the firmware volume.

12.23.1 Detailed Description

Details the location of firmware volumes that contain firmware files.

Definition at line 362 of file PiHob.h.

12.23.2 Member Data Documentation

12.23.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_FIRMWARE_VOLUME::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_FV.

Definition at line 366 of file PiHob.h.

The documentation for this struct was generated from the following file:

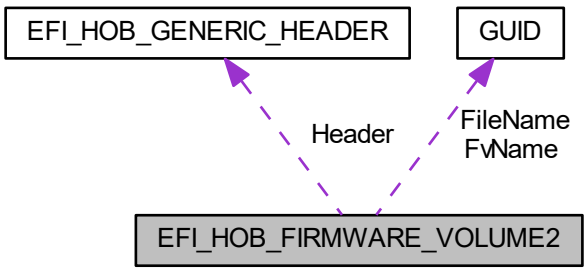
- PiHob.h

12.24 EFI_HOB_FIRMWARE_VOLUME2 Struct Reference

Details the location of a firmware volume that was extracted from a file within another firmware volume.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_FIRMWARE_VOLUME2:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_PHYSICAL_ADDRESS BaseAddress

  The physical memory-mapped base address of the firmware volume.
- UINT64 Length

  The length in bytes of the firmware volume.
- EFI_GUID FvName

  The name of the firmware volume.
- EFI_GUID FileName

  The name of the firmware file that contained this firmware volume.

12.24.1 Detailed Description

Details the location of a firmware volume that was extracted from a file within another firmware volume.

Definition at line 381 of file PiHob.h.

12.24.2 Member Data Documentation

12.24.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_FIRMWARE_VOLUME2::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_FV2.

Definition at line 385 of file PiHob.h.

The documentation for this struct was generated from the following file:

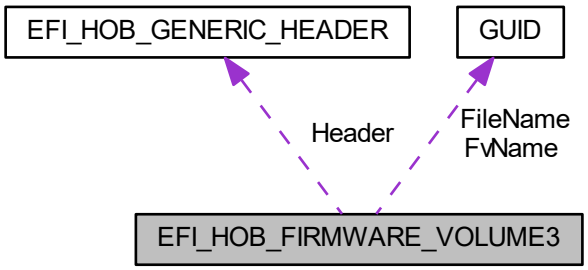
- PiHob.h

12.25 EFI_HOB_FIRMWARE_VOLUME3 Struct Reference

Details the location of a firmware volume that was extracted from a file within another firmware volume.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_FIRMWARE_VOLUME3:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header
  The HOB generic header.
- EFI_PHYSICAL_ADDRESS BaseAddress
  The physical memory-mapped base address of the firmware volume.
- UINT64 Length
  The length in bytes of the firmware volume.
- UINT32 AuthenticationStatus
  The authentication status.
- BOOLEAN ExtractedFv
  TRUE if the FV was extracted as a file within another firmware volume.
- EFI_GUID FvName
  The name of the firmware volume.
- EFI_GUID FileName
  The name of the firmware file that contained this firmware volume.

12.25.1 Detailed Description

Details the location of a firmware volume that was extracted from a file within another firmware volume.

Definition at line 408 of file PiHob.h.

12.25.2 Member Data Documentation

12.25.2.1 ExtractedFv

```
BOOLEAN EFI_HOB_FIRMWARE_VOLUME3::ExtractedFv
```

TRUE if the FV was extracted as a file within another firmware volume.

FALSE otherwise.

Definition at line 429 of file PiHob.h.

12.25.2.2 FileName

```
EFI_GUID EFI_HOB_FIRMWARE_VOLUME3::FileName
```

The name of the firmware file that contained this firmware volume.

Valid only if IsExtractedFv is TRUE.

Definition at line 439 of file PiHob.h.

12.25.2.3 FvName

```
EFI_GUID EFI_HOB_FIRMWARE_VOLUME3::FvName
```

The name of the firmware volume.

Valid only if IsExtractedFv is TRUE.

Definition at line 434 of file PiHob.h.

12.25.2.4 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_FIRMWARE_VOLUME3::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_FV3.

Definition at line 412 of file PiHob.h.

The documentation for this struct was generated from the following file:

- PiHob.h

12.26 EFI_HOB_GENERIC_HEADER Struct Reference

Describes the format and size of the data inside the HOB.

`#include <PiHob.h>`

Public Attributes

- UINT16 HobType

  Identifies the HOB data structure type.
- UINT16 HobLength

  The length in bytes of the HOB.
- UINT32 Reserved

  This field must always be set to zero.

12.26.1 Detailed Description

Describes the format and size of the data inside the HOB.

All HOBs must contain this generic HOB header.

Definition at line 36 of file PiHob.h.

The documentation for this struct was generated from the following file:

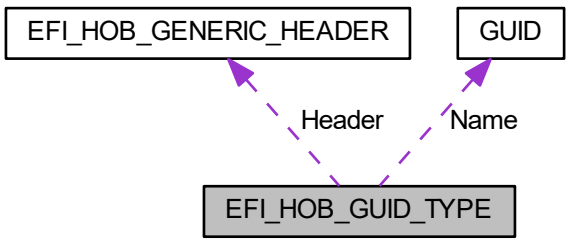
- PiHob.h

12.27 EFI_HOB_GUID_TYPE Struct Reference

Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_GUID_TYPE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header
  The HOB generic header.
- EFI_GUID Name
  A GUID that defines the contents of this HOB.

12.27.1 Detailed Description

Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID.

Definition at line 345 of file PiHob.h.

12.27.2 Member Data Documentation

12.27.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_GUID_TYPE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_GUID_EXTENSION.

Definition at line 349 of file PiHob.h.

The documentation for this struct was generated from the following file:

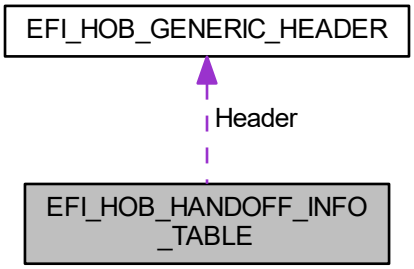
- PiHob.h

12.28 EFI_HOB_HANDOFF_INFO_TABLE Struct Reference

Contains general state information used by the HOB producer phase.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_HANDOFF_INFO_TABLE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- UINT32 Version

  The version number pertaining to the PHIT HOB definition.
- EFI_BOOT_MODE BootMode

  The system boot mode as determined during the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiMemoryTop

  The highest address location of memory that is allocated for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiMemoryBottom

  The lowest address location of memory that is allocated for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiFreeMemoryTop

  The highest address location of free memory that is currently available for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiFreeMemoryBottom

  The lowest address location of free memory that is available for use by the HOB producer phase.
- EFI_PHYSICAL_ADDRESS EfiEndOfHobList

  The end of the HOB list.

12.28.1 Detailed Description

Contains general state information used by the HOB producer phase.

This HOB must be the first one in the HOB list.

Definition at line 60 of file PiHob.h.

12.28.2 Member Data Documentation

12.28.2.1 EfiMemoryTop

```
EFI_PHYSICAL_ADDRESS EFI_HOB_HANDOFF_INFO_TABLE::EfiMemoryTop
```

The highest address location of memory that is allocated for use by the HOB producer phase.

This address must be 4-KB aligned to meet page restrictions of UEFI.

Definition at line 79 of file PiHob.h.

12.28.2.2 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_HANDOFF_INFO_TABLE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_HANDOFF.

Definition at line 64 of file PiHob.h.

12.28.2.3 Version

```
UINT32 EFI_HOB_HANDOFF_INFO_TABLE::Version
```

The version number pertaining to the PHIT HOB definition.

This value is four bytes in length to provide an 8-byte aligned entry when it is combined with the 4-byte BootMode.

Definition at line 70 of file PiHob.h.

The documentation for this struct was generated from the following file:

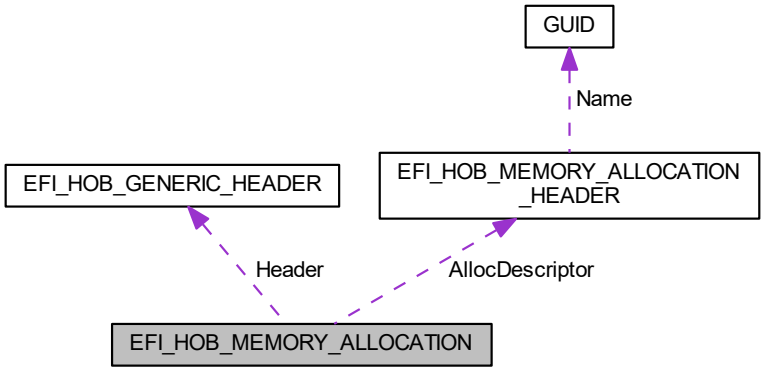
- PiHob.h

12.29 EFI_HOB_MEMORY_ALLOCATION Struct Reference

Describes all memory ranges used during the HOB producer phase that exist outside the HOB list.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER AllocDescriptor

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.

12.29.1 Detailed Description

Describes all memory ranges used during the HOB producer phase that exist outside the HOB list.

This HOB type describes how memory is used, not the physical attributes of memory.

Definition at line 144 of file PiHob.h.

12.29.2 Member Data Documentation

12.29.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 148 of file PiHob.h.

The documentation for this struct was generated from the following file:

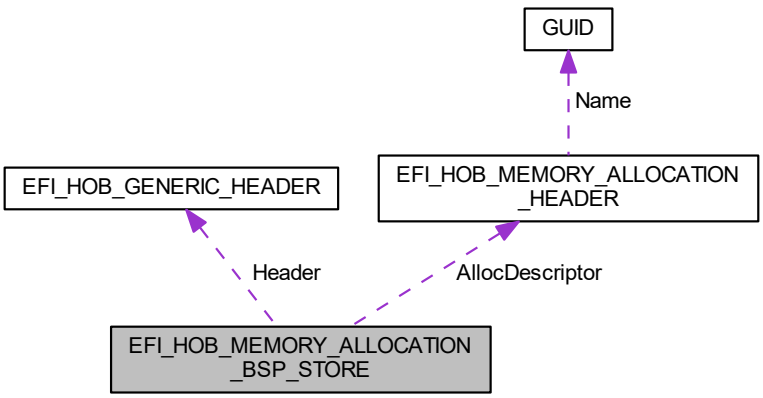
- PiHob.h

12.30 EFI_HOB_MEMORY_ALLOCATION_BSP_STORE Struct Reference

Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store").

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_BSP_STORE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER AllocDescriptor

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.

12.30.1 Detailed Description

Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store").

This HOB is valid for the Itanium processor family only register overflow store.

Definition at line 183 of file PiHob.h.

12.30.2 Member Data Documentation

12.30.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION_BSP_STORE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 187 of file PiHob.h.

The documentation for this struct was generated from the following file:

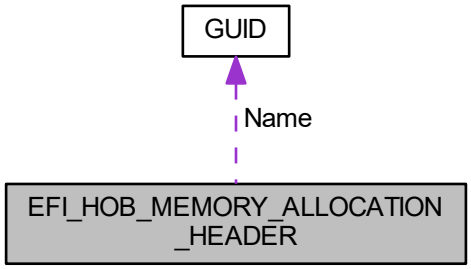
- PiHob.h

12.31 EFI_HOB_MEMORY_ALLOCATION_HEADER Struct Reference

EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_HEADER:



Public Attributes

- EFI_GUID Name

  A GUID that defines the memory allocation region's type and purpose, as well as other fields within the memory allocation HOB.
- EFI_PHYSICAL_ADDRESS MemoryBaseAddress

  The base address of memory allocated by this HOB.
- UINT64 MemoryLength

  The length in bytes of memory allocated by this HOB.
- EFI_MEMORY_TYPE MemoryType

  Defines the type of memory allocated by this HOB.
- UINT8 Reserved [4]

  Padding for Itanium processor family.

12.31.1 Detailed Description

EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation.

The type field will be used for subsequent inclusion in the UEFI memory map.

Definition at line 104 of file PiHob.h.

12.31.2 Member Data Documentation

12.31.2.1 MemoryBaseAddress

```
EFI_PHYSICAL_ADDRESS EFI_HOB_MEMORY_ALLOCATION_HEADER::MemoryBaseAddress
```

The base address of memory allocated by this HOB.

Type EFI_PHYSICAL_ADDRESS is defined in AllocatePages() in the UEFI 2.0 specification.

Definition at line 119 of file PiHob.h.

12.31.2.2 MemoryType

```
EFI_MEMORY_TYPE EFI_HOB_MEMORY_ALLOCATION_HEADER::MemoryType
```

Defines the type of memory allocated by this HOB.

The memory type definition follows the EFI_MEMORY_TYPE definition. Type EFI_MEMORY_TYPE is defined in AllocatePages() in the UEFI 2.0 specification.

Definition at line 131 of file PiHob.h.

12.31.2.3 Name

```
EFI_GUID EFI_HOB_MEMORY_ALLOCATION_HEADER::Name
```

A GUID that defines the memory allocation region's type and purpose, as well as other fields within the memory allocation HOB.

This GUID is used to define the additional data within the HOB that may be present for the memory allocation HOB. Type EFI_GUID is defined in InstallProtocolInterface() in the UEFI 2.0 specification.

Definition at line 112 of file PiHob.h.

The documentation for this struct was generated from the following file:

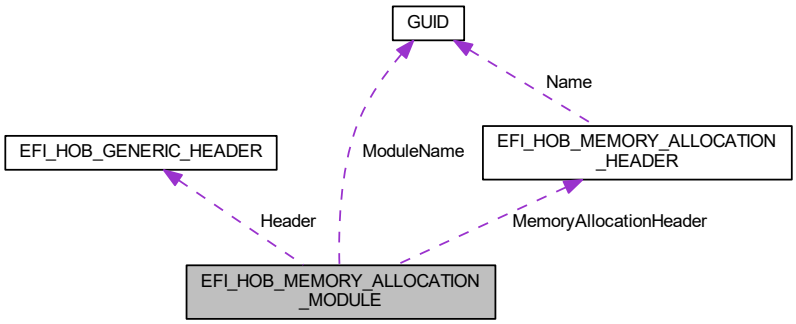
- PiHob.h

12.32 EFI_HOB_MEMORY_ALLOCATION_MODULE Struct Reference

Defines the location and entry point of the HOB consumer phase.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_MODULE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER MemoryAllocationHeader

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.
- EFI_GUID ModuleName

  The GUID specifying the values of the firmware file system name that contains the HOB consumer phase component.
- EFI_PHYSICAL_ADDRESS EntryPoint

  The address of the memory-mapped firmware volume that contains the HOB consumer phase firmware file.

12.32.1 Detailed Description

Defines the location and entry point of the HOB consumer phase.

Definition at line 198 of file PiHob.h.

12.32.2 Member Data Documentation

12.32.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION_MODULE::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 202 of file PiHob.h.

The documentation for this struct was generated from the following file:

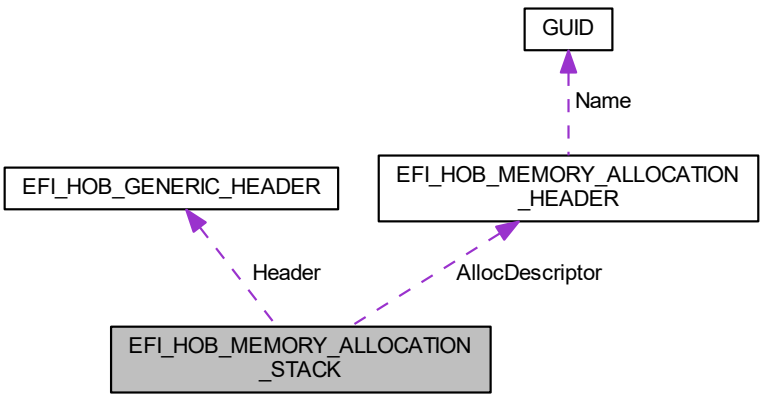
- PiHob.h

12.33 EFI_HOB_MEMORY_ALLOCATION_STACK Struct Reference

Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_ALLOCATION_STACK:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_HOB_MEMORY_ALLOCATION_HEADER AllocDescriptor

  An instance of the EFI_HOB_MEMORY_ALLOCATION_HEADER that describes the various attributes of the logical memory allocation.

12.33.1 Detailed Description

Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing.

Definition at line 165 of file PiHob.h.

12.33.2 Member Data Documentation

12.33.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_ALLOCATION_STACK::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_ALLOCATION.

Definition at line 169 of file PiHob.h.

The documentation for this struct was generated from the following file:

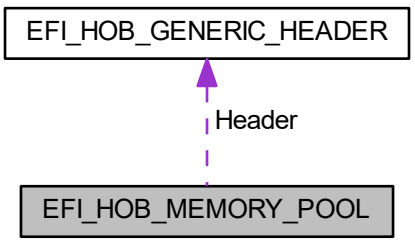
- PiHob.h

12.34 EFI_HOB_MEMORY_POOL Struct Reference

Describes pool memory allocations.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_MEMORY_POOL:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header
  The HOB generic header.

12.34.1 Detailed Description

Describes pool memory allocations.

Definition at line 467 of file PiHob.h.

12.34.2 Member Data Documentation

12.34.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_MEMORY_POOL::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_MEMORY_POOL.

Definition at line 471 of file PiHob.h.

The documentation for this struct was generated from the following file:

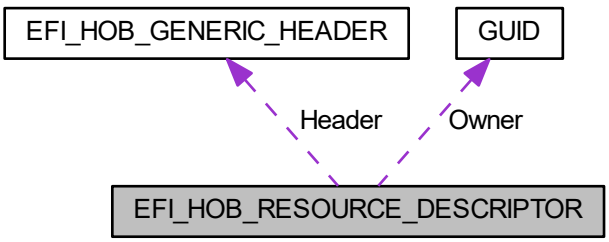
- PiHob.h

12.35 EFI_HOB_RESOURCE_DESCRIPTOR Struct Reference

Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase.

```
#include <PiHob.h>
```

Collaboration diagram for EFI_HOB_RESOURCE_DESCRIPTOR:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header.
- EFI_GUID Owner

  A GUID representing the owner of the resource.
- EFI_RESOURCE_TYPE ResourceType

  The resource type enumeration as defined by EFI_RESOURCE_TYPE.
- EFI_RESOURCE_ATTRIBUTE_TYPE ResourceAttribute

  Resource attributes as defined by EFI_RESOURCE_ATTRIBUTE_TYPE.
- EFI_PHYSICAL_ADDRESS PhysicalStart

  The physical start address of the resource region.
- UINT64 ResourceLength

  The number of bytes of the resource region.

12.35.1 Detailed Description

Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase.

Definition at line 313 of file PiHob.h.

12.35.2 Member Data Documentation

12.35.2.1 Header

```
EFI_HOB_GENERIC_HEADER EFI_HOB_RESOURCE_DESCRIPTOR::Header
```

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_RESOURCE_DESCRIPTOR.

Definition at line 317 of file PiHob.h.

12.35.2.2 Owner

```
EFI_GUID EFI_HOB_RESOURCE_DESCRIPTOR::Owner
```

A GUID representing the owner of the resource.

This GUID is used by HOB consumer phase components to correlate device ownership of a resource.

Definition at line 322 of file PiHob.h.

The documentation for this struct was generated from the following file:

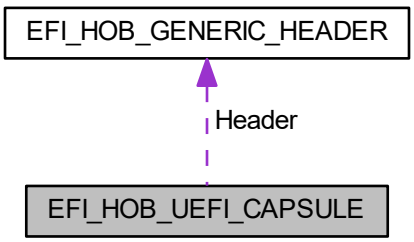
- PiHob.h

12.36 EFI_HOB_UEFI_CAPSULE Struct Reference

Each UEFI capsule HOB details the location of a UEFI capsule.

`#include <PiHob.h>`

Collaboration diagram for EFI_HOB_UEFI_CAPSULE:



Public Attributes

- EFI_HOB_GENERIC_HEADER Header

  The HOB generic header where Header.HobType = EFI_HOB_TYPE_UEFI_CAPSULE.
- EFI_PHYSICAL_ADDRESS BaseAddress

  The physical memory-mapped base address of an UEFI capsule.

12.36.1 Detailed Description

Each UEFI capsule HOB details the location of a UEFI capsule.

It includes a base address and length which is based upon memory blocks with a EFI_CAPSULE_HEADER and the associated CapsuleImageSize-based payloads. These HOB's shall be created by the PEI PI firmware sometime after the UEFI UpdateCapsule service invocation with the CAPSULE_FLAGS_POPULATE_SYSTEM_TABLE flag set in the EFI_CAPSULE_HEADER.

Definition at line 481 of file PiHob.h.

12.36.2 Member Data Documentation

12.36.2.1 BaseAddress

```
EFI_PHYSICAL_ADDRESS EFI_HOB_UEFI_CAPSULE::BaseAddress
```

The physical memory-mapped base address of an UEFI capsule.

This value is set to point to the base of the contiguous memory of the UEFI capsule. The length of the contiguous memory in bytes.

Definition at line 492 of file PiHob.h.

The documentation for this struct was generated from the following file:

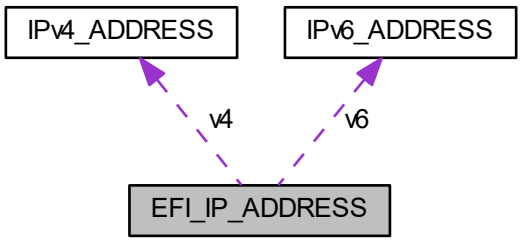
- PiHob.h

12.37 EFI_IP_ADDRESS Union Reference

16-byte buffer aligned on a 4-byte boundary.

```
#include <UefiBaseType.h>
```

Collaboration diagram for EFI_IP_ADDRESS:



12.37.1 Detailed Description

16-byte buffer aligned on a 4-byte boundary.

An IPv4 or IPv6 internet protocol address.

Definition at line 103 of file UefiBaseType.h.

The documentation for this union was generated from the following file:

- UefiBaseType.h

12.38 EFI_MAC_ADDRESS Struct Reference

32-byte buffer containing a network Media Access Control address.

`#include <UefiBaseType.h>`

12.38.1 Detailed Description

32-byte buffer containing a network Media Access Control address.

Definition at line 95 of file UefiBaseType.h.

The documentation for this struct was generated from the following file:

- UefiBaseType.h

12.39 EFI_MMRAM_DESCRIPTOR Struct Reference

Structure describing a MMRAM region and its accessibility attributes.

`#include <PiMultiPhase.h>`

Public Attributes

- EFI_PHYSICAL_ADDRESS PhysicalStart
  Designates the physical address of the MMRAM in memory.
- EFI_PHYSICAL_ADDRESS CpuStart
  Designates the address of the MMRAM, as seen by software executing on the processors.
- UINT64 PhysicalSize
  Describes the number of bytes in the MMRAM region.
- UINT64 RegionState
  Describes the accessibility attributes of the MMRAM.

12.39.1 Detailed Description

Structure describing a MMRAM region and its accessibility attributes.

Definition at line 109 of file PiMultiPhase.h.

12.39.2 Member Data Documentation

12.39.2.1 CpuStart

```
EFI_PHYSICAL_ADDRESS EFI_MMRAM_DESCRIPTOR::CpuStart
```

Designates the address of the MMRAM, as seen by software executing on the processors.

This address may or may not match PhysicalStart.

Definition at line 120 of file PiMultiPhase.h.

12.39.2.2 PhysicalStart

```
EFI_PHYSICAL_ADDRESS EFI_MMRAM_DESCRIPTOR::PhysicalStart
```

Designates the physical address of the MMRAM in memory.

This view of memory is the same as seen by I/O-based agents, for example, but it may not be the address seen by the processors.

Definition at line 115 of file PiMultiPhase.h.

12.39.2.3 RegionState

```
UINT64 EFI_MMRAM_DESCRIPTOR::RegionState
```

Describes the accessibility attributes of the MMRAM.

These attributes include the hardware state (e.g., Open/Closed/Locked), capability (e.g., cacheable), logical allocation (e.g., allocated), and pre-use initialization (e.g., needs testing/ECC initialization).

Definition at line 131 of file PiMultiPhase.h.

The documentation for this struct was generated from the following file:

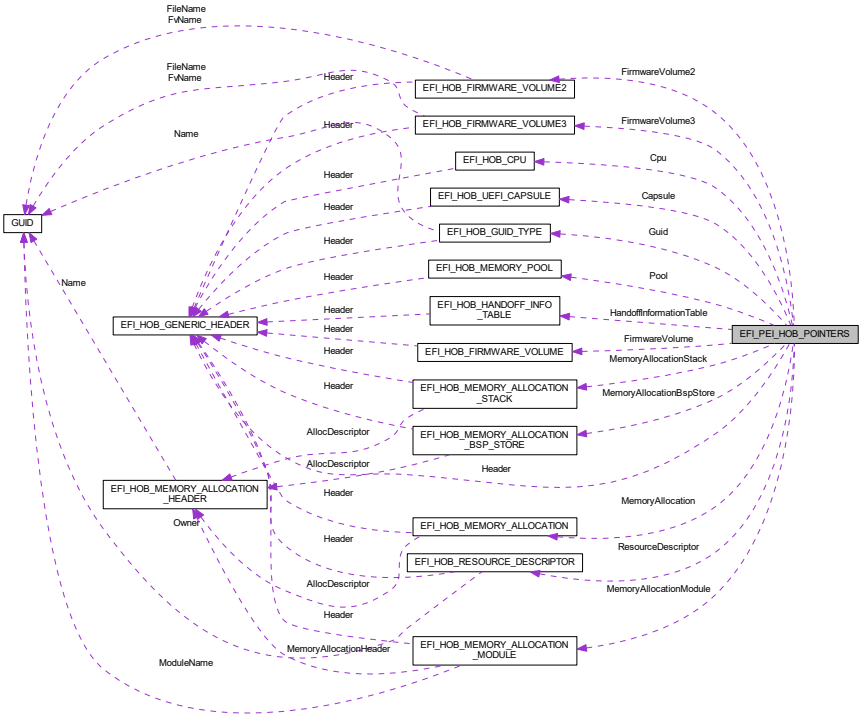
- PiMultiPhase.h

12.40 EFI_PEI_HOB_POINTERS Union Reference

Union of all the possible HOB Types.

`#include <PiHob.h>`

Collaboration diagram for EFI_PEI_HOB_POINTERS:



12.40.1 Detailed Description

Union of all the possible HOB Types.

Definition at line 499 of file PiHob.h.

The documentation for this union was generated from the following file:

- PiHob.h

12.41 EFI_TIME Struct Reference

EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047.

`#include <UefiBaseType.h>`

12.41.1 Detailed Description

EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047.

Definition at line 68 of file UefiBaseType.h.

The documentation for this struct was generated from the following file:

- UefiBaseType.h

12.42 FIVR_EXT_RAIL_CONFIG Struct Reference

Structure for V1p05/Vnn VR rail configuration.

`#include <FivrConfig.h>`

Public Attributes

- UINT32 EnabledStates: 6

  *Mask to enable the usage of external VR rail in specific S0ix or Sx states Use values from FIVR_RAIL_SX_STATE The default is **FivrRailDisabled**.*
- UINT32 Voltage: 11

  VR rail voltage value that will be used in S0i2/S0i3 states.
- UINT32 RsvdBits1: 15

  UINT32 Alignement.
- UINT32 CtrlRampTmr: 8

  This register holds the control hold off values to be used when changing the rail control for external bypass value in us.
- UINT32 SupportedVoltageStates: 4

  Mask to set the supported configuration in VR rail.
- UINT32 IccMaximum: 16

  *VR rail Icc Maximum Value Granularity of this setting is 1mA and maximal possible value is 500mA The default is **0mA** .*
- UINT32 RsvdBits2: 4

  UINT32 Alignment.

12.42.1 Detailed Description

Structure for V1p05/Vnn VR rail configuration.

Definition at line 69 of file FivrConfig.h.

12.42.2 Member Data Documentation

12.42.2.1 SupportedVoltageStates

```
UINT32 FIVR_EXT_RAIL_CONFIG::SupportedVoltageStates
```

Mask to set the supported configuration in VR rail.

Use values from FIVR_RAIL_SUPPORTED_VOLTAGE

Definition at line 97 of file FivrConfig.h.

12.42.2.2 Voltage

```
UINT32 FIVR_EXT_RAIL_CONFIG::Voltage
```

VR rail voltage value that will be used in S0i2/S0i3 states.

This value is given in 2.5mV increments (0=0mV, 1=2.5mV, 2=5mV...) The default for Vnn is set to **420 - 1050 mV**.

Definition at line 82 of file FivrConfig.h.

The documentation for this struct was generated from the following file:

- FivrConfig.h

12.43 FIVR_VCCIN_AUX_CONFIG Struct Reference

Structure for VCCIN_AUX voltage rail configuration.

```
#include <FivrConfig.h>
```

Public Attributes

- UINT8 LowToHighCurModeVolTranTime

  Transition time in microseconds from Low Current Mode Voltage to High Current Mode Voltage.
- UINT8 RetToHighCurModeVolTranTime

  Transition time in microseconds from Retention Mode Voltage to High Current Mode Voltage.
- UINT8 RetToLowCurModeVolTranTime

  Transition time in microseconds from Retention Mode Voltage to Low Current Mode Voltage.
- UINT32 OffToHighCurModeVolTranTime: 11

  Transition time in microseconds from Off (0V) to High Current Mode Voltage.

12.43.1 Detailed Description

Structure for VCCIN_AUX voltage rail configuration.

Definition at line 116 of file FivrConfig.h.

12.43.2 Member Data Documentation

12.43.2.1 LowToHighCurModeVolTranTime

```
UINT8 FIVR_VCCIN_AUX_CONFIG::LowToHighCurModeVolTranTime
```

Transition time in microseconds from Low Current Mode Voltage to High Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from the low current mode voltage and high current mode voltage. This field has 1us resolution. When value is 0 PCH will not transition VCCIN_AUX to low current mode voltage. The default is **0xC** .

Definition at line 125 of file FivrConfig.h.

12.43.2.2 OffToHighCurModeVolTranTime

```
UINT32 FIVR_VCCIN_AUX_CONFIG::OffToHighCurModeVolTranTime
```

Transition time in microseconds from Off (0V) to High Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from 0V to the high current mode voltage. This field has 1us resolution. 0 = Transition to 0V is disabled Setting this field to 0 sets VCCIN_AUX as a fixed rail that stays on in all S0 & Sx power states after initial start up on G3 exit The default is **0x96** .

Definition at line 157 of file FivrConfig.h.

12.43.2.3 RetToHighCurModeVolTranTime

```
UINT8 FIVR_VCCIN_AUX_CONFIG::RetToHighCurModeVolTranTime
```

Transition time in microseconds from Retention Mode Voltage to High Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from the retention mode voltage to high current mode voltage. This field has 1us resolution. When value is 0 PCH will not transition VCCIN_AUX to retention voltage. The default is **0x36** .

Definition at line 135 of file FivrConfig.h.

12.43.2.4 RetToLowCurModeVolTranTime

```
UINT8 FIVR_VCCIN_AUX_CONFIG::RetToLowCurModeVolTranTime
```

Transition time in microseconds from Retention Mode Voltage to Low Current Mode Voltage.

Voltage transition time required by motherboard voltage regulator when PCH changes the VCCIN_AUX regulator set point from the retention mode voltage to low current mode voltage. This field has 1us resolution. When value is 0 PCH will not transition VCCIN_AUX to retention voltage. The default is **0x2B** .

Definition at line 145 of file FivrConfig.h.

The documentation for this struct was generated from the following file:

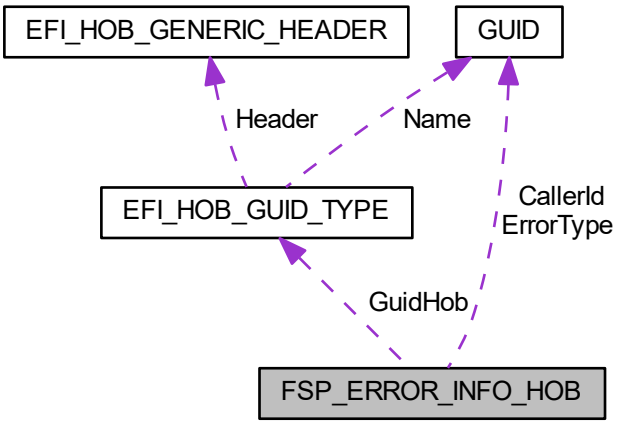
- FivrConfig.h

12.44 FSP_ERROR_INFO_HOB Struct Reference

FSP Error Information Block.

```
#include <FspErrorInfo.h>
```

Collaboration diagram for FSP_ERROR_INFO_HOB:



Public Attributes

- EFI_HOB_GUID_TYPE GuidHob

  GUID HOB header.
- EFI_STATUS_CODE_TYPE Type

  ReportStatusCode () type identifier.
- EFI_STATUS_CODE_VALUE Value

  ReportStatusCode () value.
- UINT32 Instance

  ReportStatusCode () Instance number.
- EFI_GUID CallerId

  Optional GUID which may be used to identify which internal component of the FSP was executing at the time of the error.
- EFI_GUID ErrorType

  GUID identifying the nature of the fatal error.
- UINT32 Status

  EFI_STATUS code describing the error encountered.

12.44.1 Detailed Description

FSP Error Information Block.

Definition at line 60 of file FspErrorInfo.h.

The documentation for this struct was generated from the following file:

- FspErrorInfo.h

12.45 FSPM_ARCH_CONFIG_PPI Struct Reference

This PPI provides FSP-M Arch Config PPI.

```
#include <FspmArchConfigPpi.h>
```

12.45.1 Detailed Description

This PPI provides FSP-M Arch Config PPI.

Definition at line 32 of file FspmArchConfigPpi.h.

The documentation for this struct was generated from the following file:

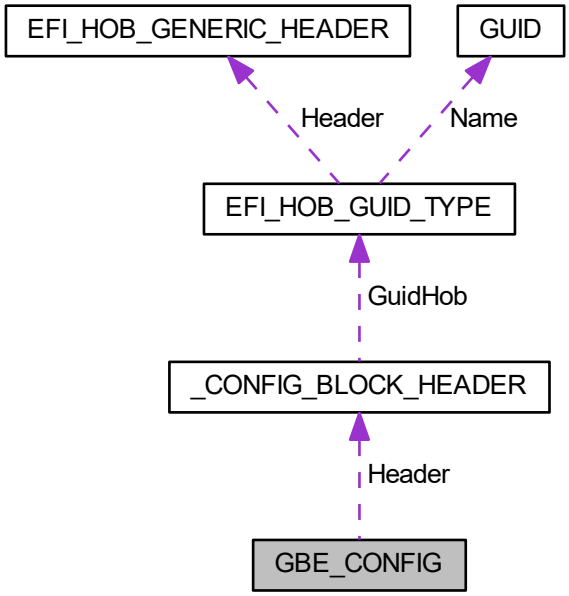
- FspmArchConfigPpi.h

12.46 GBE_CONFIG Struct Reference

PCH intergrated GBE controller configuration settings.

`#include <GbeConfig.h>`

Collaboration diagram for GBE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Enable: 1

  *Determines if enable PCH internal GBE, 0: Disable; **1: Enable**.*
- UINT32 LtrEnable: 1

  ***0: Disable**; 1: Enable LTR capabilty of PCH internal LAN.*
- UINT32 PchWOLFastSupport: 1

  Deprecated.
- UINT32 RsvdBits0: 29

  Reserved bits.

12.46.1 Detailed Description

PCH intergrated GBE controller configuration settings.

Definition at line 46 of file GbeConfig.h.

12.46.2 Member Data Documentation

12.46.2.1 Enable

```
UINT32 GBE_CONFIG::Enable
```

Determines if enable PCH internal GBE, 0: Disable; **1: Enable**.

When Enable is changed (from disabled to enabled or from enabled to disabled), it needs to set LAN Disable regsiter, which might be locked by FDSWL register. So it's recommendated to issue a global reset when changing the status for PCH Internal LAN.

Definition at line 54 of file GbeConfig.h.

12.46.2.2 PchWOLFastSupport

```
UINT32 GBE_CONFIG::PchWOLFastSupport
```

Deprecated.

Unused in MTL. Kept for compatibility. Enables bit B_PCH_ACPI_GPE0_EN_127_96_PME_B0 during PchLan↩ SxCallback in PchLanSxSmm.

Definition at line 61 of file GbeConfig.h.

The documentation for this struct was generated from the following file:

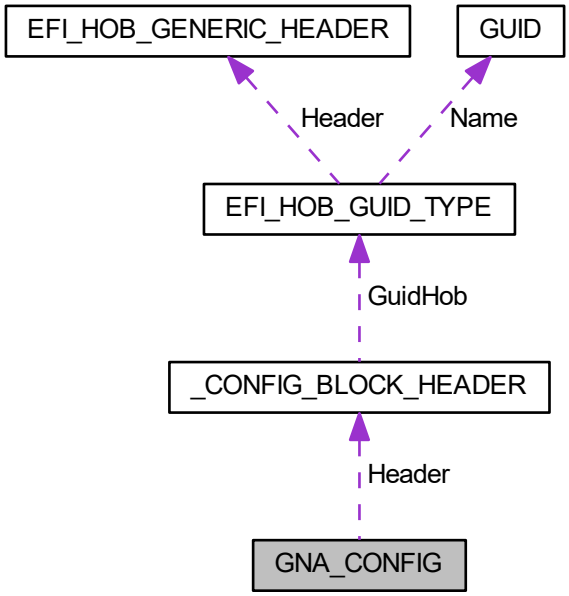
- GbeConfig.h

12.47 GNA_CONFIG Struct Reference

GNA config block for configuring GNA.

```
#include <GnaConfig.h>
```

Collaboration diagram for GNA_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT32 GnaEnable: 1

  Offset 28:0 This policy enables the GNA Device (SA Device 8) if supported.
- UINT32 RsvdBits0: 31

  Offset 28:1 :Reserved for future use.

12.47.1 Detailed Description

GNA config block for configuring GNA.

Revision 1:

- Initial version.

Definition at line 45 of file GnaConfig.h.

12.47.2 Member Data Documentation

12.47.2.1 GnaEnable

```
UINT32 GNA_CONFIG::GnaEnable
```

Offset 28:0 This policy enables the GNA Device (SA Device 8) if supported.

If FALSE, all other policies in this config block will be ignored. **1=TRUE**; 0=FALSE.

Definition at line 54 of file GnaConfig.h.

The documentation for this struct was generated from the following file:

- GnaConfig.h

12.48 GPIO_CONFIG Struct Reference

GPIO configuration structure used for pin programming.

```
#include <GpioConfig.h>
```

Public Attributes

- UINT32 PadMode: 5

  Pad Mode Pad can be set as GPIO or one of its native functions.
- UINT32 HostSoftPadOwn: 2

  Host Software Pad Ownership Set pad to ACPI mode or GPIO Driver Mode.
- UINT32 Direction: 6

  GPIO Direction Can choose between In, In with inversion, Out, both In and Out, both In with inversion and out or disabling both.
- UINT32 OutputState: 2

  Output State Set Pad output value.
- UINT32 InterruptConfig: 9

  GPIO Interrupt Configuration Set Pad to cause one of interrupts (IOxAPIC/SCI/SMI/NMI).
- UINT32 PowerConfig: 8

  GPIO Power Configuration.
- UINT32 ElectricalConfig: 9

  GPIO Electrical Configuration This setting controls pads termination and voltage tolerance.
- UINT32 LockConfig: 4

  GPIO Lock Configuration This setting controls pads lock.
- UINT32 OtherSettings: 2

  Additional GPIO configuration Refer to definition of GPIO_OTHER_CONFIG for supported settings.
- UINT32 RsvdBits: 17

  Reserved bits for future extension.

12.48.1 Detailed Description

GPIO configuration structure used for pin programming.

Structure contains fields that can be used to configure pad.

Definition at line 55 of file GpioConfig.h.

12.48.2 Member Data Documentation

12.48.2.1 Direction

`UINT32 GPIO_CONFIG::Direction`

GPIO Direction Can choose between In, In with inversion, Out, both In and Out, both In with inversion and out or disabling both.

Refer to definition of GPIO_DIRECTION for supported settings.

Definition at line 76 of file GpioConfig.h.

12.48.2.2 ElectricalConfig

`UINT32 GPIO_CONFIG::ElectricalConfig`

GPIO Electrical Configuration This setting controls pads termination and voltage tolerance.

Refer to definition of GPIO_ELECTRICAL_CONFIG for supported settings.

Definition at line 102 of file GpioConfig.h.

12.48.2.3 HostSoftPadOwn

`UINT32 GPIO_CONFIG::HostSoftPadOwn`

Host Software Pad Ownership Set pad to ACPI mode or GPIO Driver Mode.

Refer to definition of GPIO_HOSTSW_OWN.

Definition at line 70 of file GpioConfig.h.

12.48.2.4 InterruptConfig

`UINT32 GPIO_CONFIG::InterruptConfig`

GPIO Interrupt Configuration Set Pad to cause one of interrupts (IOxAPIC/SCI/SMI/NMI).

This setting is applicable only if GPIO is in GpioMode with input enabled. Refer to definition of GPIO_INT_CONFIG for supported settings.

Definition at line 90 of file GpioConfig.h.

12.48.2.5 LockConfig

```
UINT32 GPIO_CONFIG::LockConfig
```

GPIO Lock Configuration This setting controls pads lock.

Refer to definition of GPIO_LOCK_CONFIG for supported settings.

Definition at line 108 of file GpioConfig.h.

12.48.2.6 OutputState

```
UINT32 GPIO_CONFIG::OutputState
```

Output State Set Pad output value.

Refer to definition of GPIO_OUTPUT_STATE for supported settings. This setting takes place when output is enabled.

Definition at line 83 of file GpioConfig.h.

12.48.2.7 PadMode

```
UINT32 GPIO_CONFIG::PadMode
```

Pad Mode Pad can be set as GPIO or one of its native functions.

When in native mode setting Direction (except Inversion), OutputState, InterruptConfig, Host Software Pad Ownership and OutputStateLock are unnecessary. Refer to definition of GPIO_PAD_MODE. Refer to EDS for each native mode according to the pad.

Definition at line 64 of file GpioConfig.h.

12.48.2.8 PowerConfig

```
UINT32 GPIO_CONFIG::PowerConfig
```

GPIO Power Configuration.

This setting controls Pad Reset Configuration. Refer to definition of GPIO_RESET_CONFIG for supported settings.

Definition at line 96 of file GpioConfig.h.

The documentation for this struct was generated from the following file:

- GpioConfig.h

12.49 GUID Struct Reference

128 bit buffer containing a unique identifier value.

```
#include <Base.h>
```

12.49.1 Detailed Description

128 bit buffer containing a unique identifier value.

Unless otherwise specified, aligned on a 64 bit boundary.

Definition at line 213 of file Base.h.

The documentation for this struct was generated from the following file:

- Base.h

12.50 HDA_LINK_DMIC Struct Reference

HD Audio DMIC Interface Policies.

```
#include <HdAudioConfig.h>
```

Public Attributes

- UINT32 Enable: 1

  *HDA DMIC interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*
- UINT32 DmicClockSelect: 2

  *DMIC link clock select: **0: Both**, 1: ClkA, 2: ClkB; default is "Both".*
- HDA_DMIC_PIN_MUX PinMux

  Pin mux configuration.

12.50.1 Detailed Description

HD Audio DMIC Interface Policies.

Definition at line 134 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.51 HDA_LINK_HDA Struct Reference

HD Audio Link Policies.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT32 Enable: 1

  *HDA interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*
- UINT8 SdiEnable [PCH_MAX_HDA_SDI]

  *HDA SDI signal enable. When enabled related SDI pins will be switched to appropriate native mode: **0: Disable**; 1: Enable.*
- UINT8 Reserved [(4 -(PCH_MAX_HDA_SDI % 4)) % 4]

  Padding for SDI enable table.

12.51.1 Detailed Description

HD Audio Link Policies.

Definition at line 124 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.52 HDA_LINK_SNDW Struct Reference

HD Audio SNDW Interface Policies.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT32 Enable: 1

  *HDA SNDW interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*

12.52.1 Detailed Description

HD Audio SNDW Interface Policies.

Definition at line 152 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.53 HDA_LINK_SSP Struct Reference

HD Audio SSP Interface Policies.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT32 Enable: 1
  *HDA SSP interface enable. When enabled related pins will be switched to native mode: **0: Disable**; 1: Enable.*

12.53.1 Detailed Description

HD Audio SSP Interface Policies.

Definition at line 144 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.54 HDA_VERB_TABLE_HEADER Struct Reference

Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands.

`#include <HdAudioConfig.h>`

Public Attributes

- UINT16 VendorId
  Codec Vendor ID.
- UINT16 DeviceId
  Codec Device ID.
- UINT8 RevisionId
  Revision ID of the codec. 0xFF matches any revision.
- UINT8 SdiNum
  SDI number, 0xFF matches any SDI.
- UINT16 DataDwords
  Number of data DWORDs following the header.

12.54.1 Detailed Description

Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands.

Definition at line 91 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

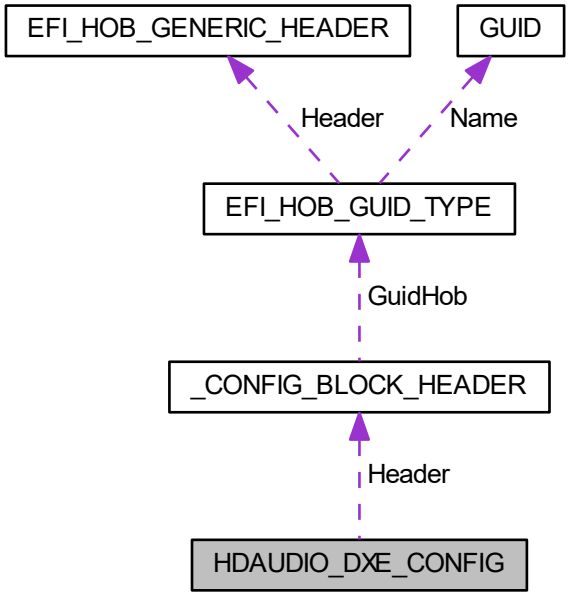
- HdAudioConfig.h

12.55 HDAUDIO_DXE_CONFIG Struct Reference

This structure contains the DXE policies which are related to HD Audio device (cAVS).

`#include <HdAudioConfig.h>`

Collaboration diagram for HDAUDIO_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.

- HDAUDIO_SNDW_CONFIG SndwConfig [PCH_MAX_HDA_SNDW_LINK_NUM]

  SNDW configuration for exposed via SNDW ACPI tables:

- UINT32 DspFeatureMask

  Bitmask of supported DSP features: [BIT0] - WoV; [BIT1] - BT Sideband; [BIT2] - Codec VAD; [BIT5] - BT Intel HFP; [BIT6] - BT Intel A2DP [BIT7] - DSP based speech pre-processing disabled; [BIT8] - 0: Intel WoV, 1: Windows Voice Activation [BIT9] - BT Intel Low Energy;[BIT9] - 0 BT Intel Low Energy is not supported , 1: BT Intel Low Energy is supported Default is ***zero****.*

- HDAUDIO_DISC_BT_OFFLOAD HdaDiscBtOffload

  Discrete BT HCI Audio Offload Support.

12.55.1 Detailed Description

This structure contains the DXE policies which are related to HD Audio device (cAVS).

Revision 1:

- Inital version. **Revision 2:**
- Add IoControlEnabled **Revision 3:**
- Remove IoControlEnabled

Definition at line 268 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

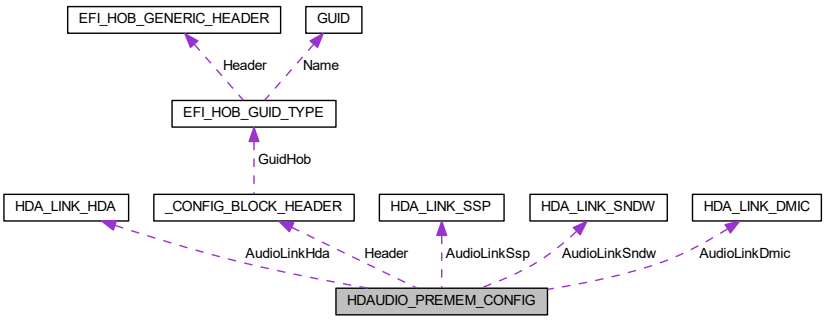
- HdAudioConfig.h

12.56 HDAUDIO_PREMEM_CONFIG Struct Reference

This structure contains the premem policies which are related to HD Audio device (cAVS).

`#include <HdAudioConfig.h>`

Collaboration diagram for HDAUDIO_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Enable: 1

  *Intel HD Audio (Azalia) enablement: 0: Disable, **1: Enable***
- UINT32 DspEnable: 1

  *DSP enablement: 0: Disable; **1: Enable***
- UINT32 VcType: 1

  *Virtual Channel Type Select: **0: VC0**, 1: VC1.*

- UINT32 DspUaaCompliance: 1

  *Universal Audio Architecture compliance for DSP enabled system: **0: Not-UAA Compliant (Intel SST driver supported only)**, 1: UAA Compliant (HDA Inbox driver or SST driver supported)*
- UINT32 IDispLinkFrequency: 4

  *iDisp-Link frequency (PCH_HDAUDIO_LINK_FREQUENCY enum): **4: 96MHz**, 3: 48MHz*
- UINT32 IDispLinkTmode: 3

  *iDisp-Link T-Mode (PCH_HDAUDIO_IDISP_TMODE enum): 0: 2T, 1: 1T, 2: 4T, **3: 8T**, 4: 16T*
- UINT32 IDispCodecDisconnect: 1

  *iDisplay Audio Codec disconnection, **0: Not disconnected, enumerable**; 1: Disconnected SDI, not enumerable*
- UINT32 PowerGatingSupported: 1

  *Power Gating supported: **0: Not supported**, 1: Supported.*
- UINT32 SndwMultilaneEnable: 2

  *Sndw0 Multiline enablement: **0: Disable**, 1: Two lines enabled, 2: Three lines enabled, 3: Four Lines enabled.*
- UINT32 RsvdBits: 17

  Reserved bits 0.
- HDA_LINK_HDA AudioLinkHda

  Audio Link Mode configuration bitmask.
- HDA_LINK_DMIC AudioLinkDmic [PCH_MAX_HDA_DMIC_LINK_NUM]

  *DMIC link enablement: 0: Disable; **1: Enable**.*
- HDA_LINK_SSP AudioLinkSsp [PCH_MAX_HDA_SSP_LINK_NUM]

  *I2S/SSP link enablement: **0: Disable**; 1: Enable.*
- HDA_LINK_SNDW AudioLinkSndw [PCH_MAX_HDA_SNDW_LINK_NUM]

  *SoundWire link enablement: **0: Disable**; 1: Enable.*
- UINT16 ResetWaitTimer

  ***(Test)** The delay timer after Azalia reset, the value is number of microseconds. Default is **600**.*
- UINT8 Rsvd0 [2]

  Reserved bytes, align to multiple 4.
- UINT32 SubSystemIds

  Value for SID and SVID. If its set to 0 then default value is used.
- HDAUDIO_DISC_BT_OFFLOAD HdaDiscBtOffload

  Discrete BT HCI Audio Offload Support.

12.56.1 Detailed Description

This structure contains the prememem policies which are related to HD Audio device (cAVS).

Revision 1:

- Inital version. **Revision 2:**
- Add DmicClockSelect **Revision 3:**
- Add AudioFpgaSocVer **Revision 4:**
- Remove AudioFpgaSocVer **Revision 5:**
- Add HdaDiscBtOffload

Definition at line 199 of file HdAudioConfig.h.

12.56.2 Member Data Documentation

12.56.2.1 AudioLinkDmic

HDA_LINK_DMIC HDAUDIO_PREMEM_CONFIG::AudioLinkDmic[PCH_MAX_HDA_DMIC_LINK_NUM]

DMIC link enablement: 0: Disable; **1: Enable**.

DMIC0 LKF: Muxed with SNDW2/SNDW4.

Definition at line 227 of file HdAudioConfig.h.

12.56.2.2 AudioLinkHda

HDA_LINK_HDA HDAUDIO_PREMEM_CONFIG::AudioLinkHda

Audio Link Mode configuration bitmask.

Allows to configure enablement of the following interfaces: HDA-Link, DMIC, SSP, SoundWire. HDA-Link enablement: 0: Disable; **1: Enable**.

Definition at line 222 of file HdAudioConfig.h.

12.56.2.3 AudioLinkSndw

HDA_LINK_SNDW HDAUDIO_PREMEM_CONFIG::AudioLinkSndw[PCH_MAX_HDA_SNDW_LINK_NUM]

SoundWire link enablement: **0: Disable**; 1: Enable.

SNDW2 LKF: Muxed with DMIC0/DMIC1. SNDW3 LKF: Muxed with DMIC1. SNDW4 LKF: Muxed with DMIC0.

Definition at line 241 of file HdAudioConfig.h.

12.56.2.4 AudioLinkSsp

HDA_LINK_SSP HDAUDIO_PREMEM_CONFIG::AudioLinkSsp[PCH_MAX_HDA_SSP_LINK_NUM]

I2S/SSP link enablement: **0: Disable**; 1: Enable.

SSP0/1 LKF: Muxed with HDA.

Note

Since the I2S/SSP2 pin set contains pads which are also used for CNVi purpose, enabling AudioLinkSsp2 is exclusive with CNVi is present.

Definition at line 234 of file HdAudioConfig.h.

The documentation for this struct was generated from the following file:

- HdAudioConfig.h

12.57 HSIO_PARAMETERS Struct Reference

This structure describes USB3 Port N configuration parameters.

`#include <Usb3HsioConfig.h>`

Public Attributes

- UINT8 HsioTxDownscaleAmp

  USB 3.0 TX Output Downscale Amplitude Adjustment (orate01margin) HSIO_TX_DWORD8[21:16] ***Default = 00h***
- UINT8 HsioTxDeEmph

  USB 3.0 TX Output -3.5dB De-Emphasis Adjustment Setting (ow2tapgen2deemph3p5) HSIO_TX_DWORD5[21:16] ***Default = 29h*** *(approximately -3.5dB De-Emphasis)*
- UINT8 HsioCtrlAdaptOffsetCfg

  Signed Magnatude number added to the CTLE code.
- UINT8 HsioFilterSelN

  LFPS filter select for n (filter_sel_n_2_0) HSIO_RX_DWORD51 [29:27] 0h:1.6ns 1h:2.4ns 2h:3.2ns 3h:4.0ns 4h:4.8ns 5h:5.6ns 6h:6.4ns ***Default = 0h***
- UINT8 HsioFilterSelP

  LFPS filter select for p (filter_sel_p_2_0) HSIO_RX_DWORD51 [26:24] 0h:1.6ns 1h:2.4ns 2h:3.2ns 3h:4.0ns 4h:4.8ns 5h:5.6ns 6h:6.4ns ***Default = 0h***
- UINT8 HsioOlfpsCfgPullUpDwnRes

  Controls the input offset (olfpscfgpullupdwnres_sus_usb_2_0) HSIO_RX_DWORD51 [2:0] 000 Prohibited 001 45K 010 Prohibited 011 31K 100 36K 101 36K 110 36K 111 36K ***Default = 3h***
- UINT8 HsioTxDeEmphEnable

  Enable the write to USB 3.0 TX Output -3.5dB De-Emphasis Adjustment, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxDownscaleAmpEnable

  Enable the write to USB 3.0 TX Output Downscale Amplitude Adjustment, ***0: Disable****; 1: Enable.*
- UINT8 HsioCtrlAdaptOffsetCfgEnable

  Enable the write to Signed Magnatude number added to the CTLE code, ***0: Disable****; 1: Enable.*
- UINT8 HsioFilterSelNEnable

  Enable the write to LFPS filter select for n, ***0: Disable****; 1: Enable.*
- UINT8 HsioFilterSelPEnable

  Enable the write to LFPS filter select for p, ***0: Disable****; 1: Enable.*
- UINT8 HsioOlfpsCfgPullUpDwnResEnable

  Enable the write to olfpscfgpullupdwnres, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate3UniqTran

  USB 3.0 TX Output - Unique Transition Bit Scale for rate 3 (rate3UniqTranScale) HSIO_TX_DWORD9[6:0] ***Default = 4Ch***
- UINT8 HsioTxRate2UniqTran

  USB 3.0 TX Output -Unique Transition Bit Scale for rate 2 (rate2UniqTranScale) HSIO_TX_DWORD9[14:8] ***Default = 4Ch***
- UINT8 HsioTxRate1UniqTran

  USB 3.0 TX Output - Unique Transition Bit Scale for rate 1 (rate1UniqTranScale) HSIO_TX_DWORD9[22:16] ***Default = 4Ch***
- UINT8 HsioTxRate0UniqTran

  USB 3.0 TX Output - Unique Transition Bit Scale for rate 0 (rate0UniqTranScale) HSIO_TX_DWORD9[30:24] ***Default = 4Ch***
- UINT8 HsioTxRate3UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 3, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate2UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 2, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate1UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 1, ***0: Disable****; 1: Enable.*
- UINT8 HsioTxRate0UniqTranEnable

  Enable the write to USB 3.0 TX Unique Transition Bit Mode for rate 0, ***0: Disable****; 1: Enable.*

12.57.1 Detailed Description

This structure describes USB3 Port N configuration parameters.

Definition at line 55 of file Usb3HsioConfig.h.

12.57.2 Member Data Documentation

12.57.2.1 HsioCtrlAdaptOffsetCfg

```
UINT8 HSIO_PARAMETERS::HsioCtrlAdaptOffsetCfg
```

Signed Magnatude number added to the CTLE code.

(ctle_adapt_offset_cfg_4_0) HSIO_RX_DWORD25 [20:16] Ex: -1 – 1_0001. +1: 0_0001 **Default = 0h**

Definition at line 74 of file Usb3HsioConfig.h.

The documentation for this struct was generated from the following file:

- Usb3HsioConfig.h

12.58 I2C_PIN_MUX Struct Reference

I2C signals pin muxing settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 Sda

  SDA Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_I2Cx_SDA_∗.
- UINT32 Scl

  SCL Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_I2Cx_SCL_∗.

12.58.1 Detailed Description

I2C signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_I2Cx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 212 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

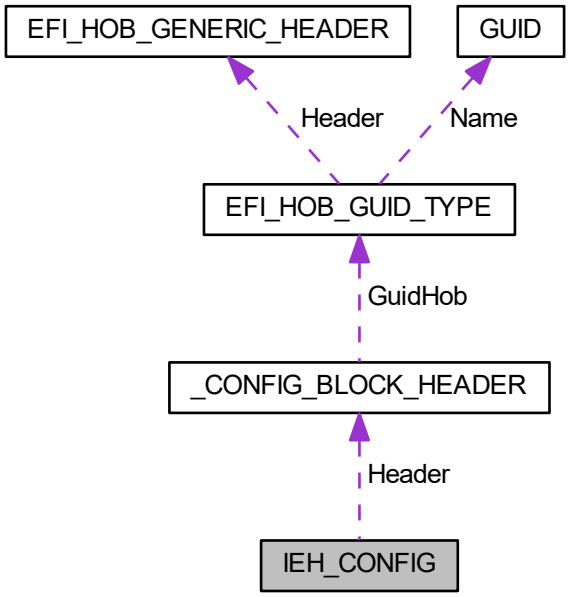
- SerialIoDevices.h

12.59 IEH_CONFIG Struct Reference

The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler.

`#include <IehConfig.h>`

Collaboration diagram for IEH_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 Mode: 1
  *IEH mode **0: Bypass Mode**; 1: Enable.*
- UINT32 RsvdBits0: 31
  Reserved bits.

12.59.1 Detailed Description

The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler.

Definition at line 53 of file IehConfig.h.

The documentation for this struct was generated from the following file:

- IehConfig.h

12.60 IOM_AUX_ORI_PAD_CONFIG Struct Reference

The IOM_AUX_ORI_PAD_CONFIG describes IOM TypeC port map GPIO pin.

`#include <TcssPeiConfig.h>`

Public Attributes

- UINT32 GpioAuxBiasCtrlPuP

  GPIO Aux Bias Ctrl Pull Up Pin number that is for IOM indicate the pull up pin from TypeC port.
- UINT32 GpioAuxBiasCtrlPdN

  GPIO Aux Bais Ctrl Pull Down Pin number that is for IOM indicate the pull down pin from TypeC port.
- UINT32 GpioIsoCtrlPin

  GPIO ISO control pin GPIO number that is IOM indicate the AUX active from SOC.

12.60.1 Detailed Description

The IOM_AUX_ORI_PAD_CONFIG describes IOM TypeC port map GPIO pin.

Those GPIO setting for DP Aux Orientation Bias Control when the TypeC port didn't have re-timer. IOM needs know Pull-Up and Pull-Down pin for Bias control

Definition at line 61 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiConfig.h

12.61 IOM_INTERFACE_CONFIG Struct Reference

The IOM_EC_INTERFACE_CONFIG block describes interaction between BIOS and IOM-EC.

`#include <TcssPeiConfig.h>`

Public Attributes

- UINT32 VccSt

  IOM VCCST request. (Not equal to actual VCCST value)
- UINT32 UsbOverride

  IOM to override USB connection.
- UINT32 D3ColdEnable

  Enable/disable D3 Cold support in TCSS.
- UINT32 D3HotEnable

  Enable/disable D3 Hot support in TCSS.

12.61.1 Detailed Description

The IOM_EC_INTERFACE_CONFIG block describes interaction between BIOS and IOM-EC.

Definition at line 71 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

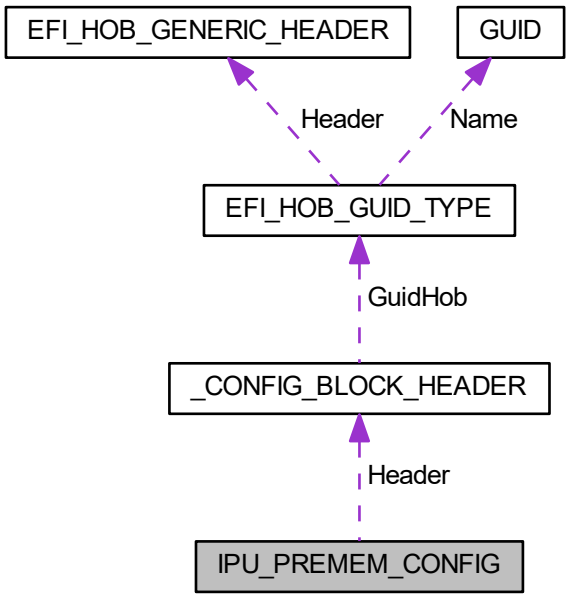
- TcssPeiConfig.h

12.62 IPU_PREMEM_CONFIG Struct Reference

IPU PreMem configuration
Revision 1:

```
#include <IpuPreMemConfig.h>
```

Collaboration diagram for IPU_PREMEM_CONFIG:



Public Attributes

- UINT8 IpuEnable

  Config Block Header.
- UINT8 ImguClkOutEn [GPIO_IMGUCLK_NUMBER_OF_PINS]

  It enable the IMGU CLKOUT.
- UINT8 Rsvd [5]

  Reserved for future use.

12.62.1 Detailed Description

IPU PreMem configuration
Revision 1:

- Initial version.

Definition at line 62 of file IpuPreMemConfig.h.

12.62.2 Member Data Documentation

12.62.2.1 ImguClkOutEn

```
UINT8 IPU_PREMEM_CONFIG::ImguClkOutEn[GPIO_IMGUCLK_NUMBER_OF_PINS]
```

It enable the IMGU CLKOUT.

TRUE FALSE

Definition at line 76 of file IpuPreMemConfig.h.

12.62.2.2 IpuEnable

```
UINT8 IPU_PREMEM_CONFIG::IpuEnable
```

Config Block Header.

(Test) It enables the SA IPU Device if supported and not fused off. If FALSE, all other policies in this config block will be ignored. **1=TRUE**; 0=FALSE.

Definition at line 70 of file IpuPreMemConfig.h.

The documentation for this struct was generated from the following file:

- IpuPreMemConfig.h

12.63 IPv4_ADDRESS Struct Reference

4-byte buffer.

```
#include <Base.h>
```

12.63.1 Detailed Description

4-byte buffer.

An IPv4 internet protocol address.

Definition at line 223 of file Base.h.

The documentation for this struct was generated from the following file:

- Base.h

12.64 IPv6_ADDRESS Struct Reference

16-byte buffer.

```
#include <Base.h>
```

12.64.1 Detailed Description

16-byte buffer.

An IPv6 internet protocol address.

Definition at line 230 of file Base.h.

The documentation for this struct was generated from the following file:

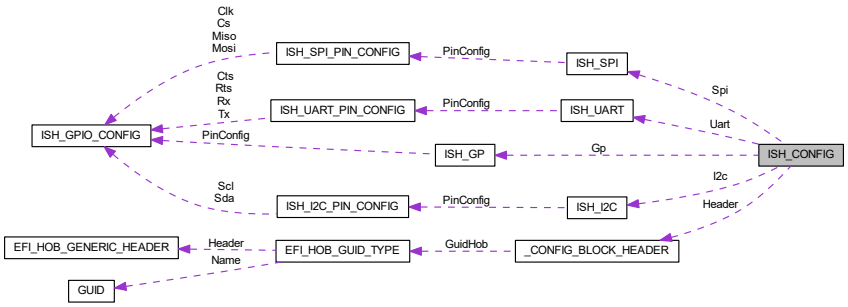
- Base.h

12.65 ISH_CONFIG Struct Reference

The ISH_CONFIG block describes Integrated Sensor Hub device.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PdtUnlock: 1

  *ISH PDT Unlock Msg: **0: False** 1: True.*
- UINT32 MsiInterrupt: 1

  *ISH MSI Interrupts: **0: False** 1: True.*
- UINT32 RsvdBits0: 30

  Reserved Bits.

12.65.1 Detailed Description

The ISH_CONFIG block describes Integrated Sensor Hub device.

Definition at line 155 of file IshConfig.h.

The documentation for this struct was generated from the following file:

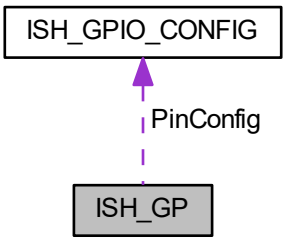
- IshConfig.h

12.66 ISH_GP Struct Reference

Struct contains GPIO pins assigned and signal settings of GP.

`#include <IshConfig.h>`

Collaboration diagram for ISH_GP:



Public Attributes

- UINT32 Enable: 1

  *ISH GP GPIO pins assigned: **0: False** 1: True.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.66.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of GP.

Definition at line 146 of file IshConfig.h.

The documentation for this struct was generated from the following file:

- IshConfig.h

12.67 ISH_GPIO_CONFIG Struct Reference

ISH GPIO settings.

`#include <IshConfig.h>`

Public Attributes

- UINT32 PinMux
  GPIO signals pin muxing settings.
- UINT32 PadTermination
  GPIO Pads Internal Termination.

12.67.1 Detailed Description

ISH GPIO settings.

Definition at line 68 of file IshConfig.h.

12.67.2 Member Data Documentation

12.67.2.1 PadTermination

`UINT32 ISH_GPIO_CONFIG::PadTermination`

GPIO Pads Internal Termination.

For more information please see Platform Design Guide. Check GPIO_ELECTRICAL_CONFIG for reference

Definition at line 80 of file IshConfig.h.

12.67.2.2 PinMux

```
UINT32 ISH_GPIO_CONFIG::PinMux
```

GPIO signals pin muxing settings.

If signal can be enable only on a single pin then this parameter should be set to 0. Refer to GPIO_∗_MUXI↩ NG_ISH_∗x_∗ in GpioPins∗.h for supported settings on a given platform GPIO Pin mux configuration. Refer to GPIO_∗_MUXING_ISH_∗x_MOSI_∗

Definition at line 74 of file IshConfig.h.

The documentation for this struct was generated from the following file:

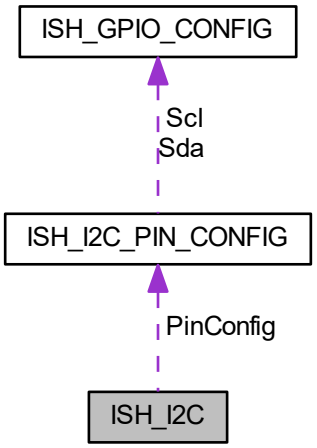
- IshConfig.h

12.68 ISH_I2C Struct Reference

Struct contains GPIO pins assigned and signal settings of I2C.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_I2C:



Public Attributes

- UINT32 Enable: 1

  ISH I2C GPIO pins assigned: ***0: False*** *1: True.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.68.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of I2C.

Definition at line 137 of file IshConfig.h.

The documentation for this struct was generated from the following file:

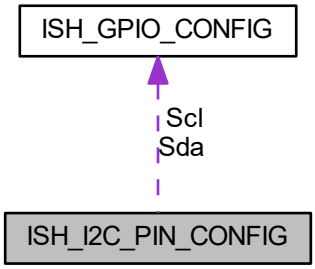
- IshConfig.h

12.69 ISH_I2C_PIN_CONFIG Struct Reference

I2C signals settings.

`#include <IshConfig.h>`

Collaboration diagram for ISH_I2C_PIN_CONFIG:



Public Attributes

- ISH_GPIO_CONFIG Sda
  SDA Pin configuration.
- ISH_GPIO_CONFIG Scl
  SCL Pin configuration.

12.69.1 Detailed Description

I2C signals settings.

Definition at line 108 of file IshConfig.h.

The documentation for this struct was generated from the following file:

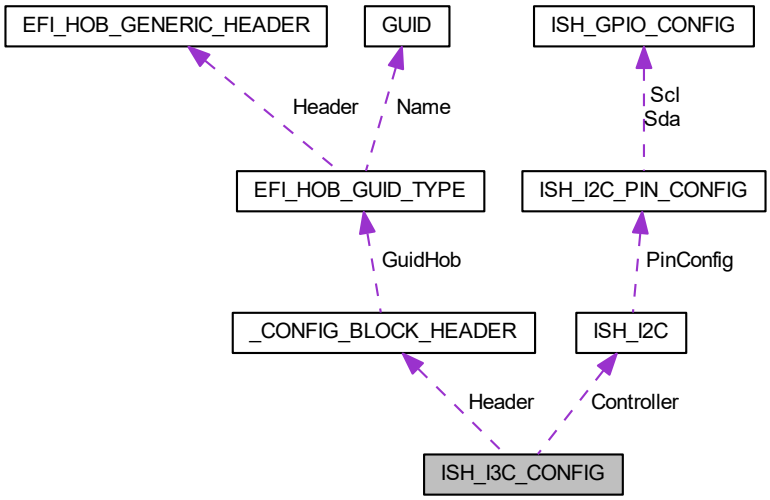
- IshConfig.h

12.70 ISH_I3C_CONFIG Struct Reference

The ISH_I3C_CONFIG block describes I3C in ISH device.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_I3C_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- ISH_I3C Controller

  ISH I3C Controller.

12.70.1 Detailed Description

The ISH_I3C_CONFIG block describes I3C in ISH device.

Definition at line 171 of file IshConfig.h.

The documentation for this struct was generated from the following file:

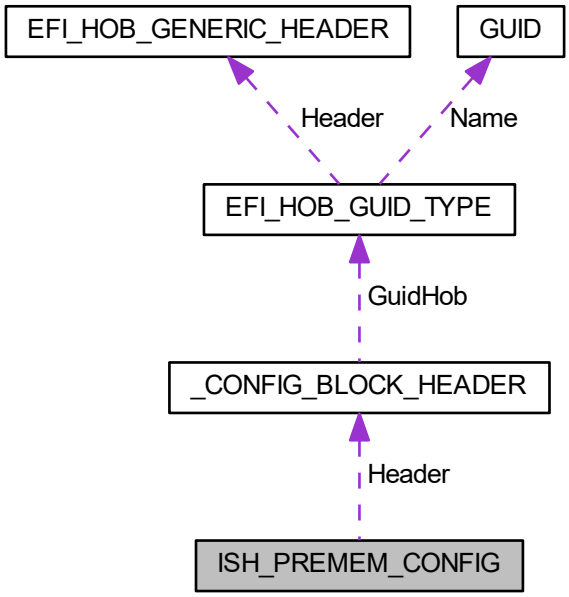
- IshConfig.h

12.71 ISH_PREMEM_CONFIG Struct Reference

Premem Policy for Integrated Sensor Hub device.

`#include <IshConfig.h>`

Collaboration diagram for ISH_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Enable: 1

  *ISH Controler 0: Disable; **1: Enable**.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.71.1 Detailed Description

Premem Policy for Integrated Sensor Hub device.

Definition at line 180 of file IshConfig.h.

12.71.2 Member Data Documentation

12.71.2.1 Enable

`UINT32 ISH_PREMEM_CONFIG::Enable`

ISH Controler 0: Disable; **1: Enable**.

For Desktop sku, the ISH POR should be disabled. **0:Disable** .

Definition at line 186 of file IshConfig.h.

The documentation for this struct was generated from the following file:

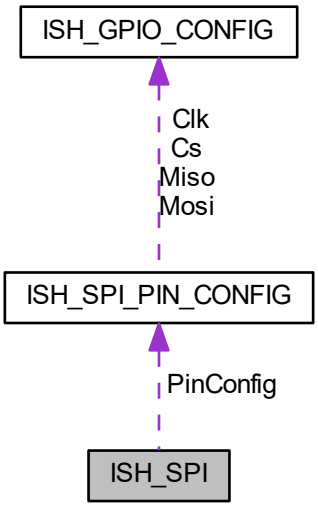
- IshConfig.h

12.72 ISH_SPI Struct Reference

Struct contains GPIO pins assigned and signal settings of SPI.

`#include <IshConfig.h>`

Collaboration diagram for ISH_SPI:



Public Attributes

- UINT8 Enable

  *ISH SPI GPIO pins assigned: **0: False** 1: True.*
- UINT8 CsEnable [PCH_MAX_ISH_SPI_CS_PINS]

  *ISH SPI CS pins assigned: **0: False** 1: True.*
- UINT16 RsvdField0

  Reserved field.

12.72.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of SPI.

Definition at line 117 of file IshConfig.h.

The documentation for this struct was generated from the following file:

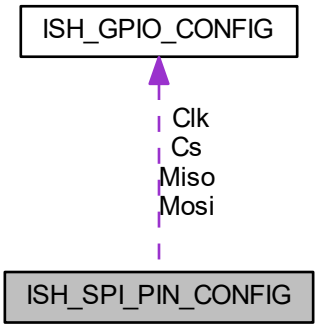
- IshConfig.h

12.73 ISH_SPI_PIN_CONFIG Struct Reference

SPI signals settings.

`#include <IshConfig.h>`

Collaboration diagram for ISH_SPI_PIN_CONFIG:



Public Attributes

- ISH_GPIO_CONFIG Mosi

  MOSI Pin configuration.
- ISH_GPIO_CONFIG Miso

  MISO Pin configuration.
- ISH_GPIO_CONFIG Clk

  CLK Pin configuration.
- ISH_GPIO_CONFIG Cs [PCH_MAX_ISH_SPI_CS_PINS]

  CS Pin configuration.

12.73.1 Detailed Description

SPI signals settings.

Definition at line 86 of file IshConfig.h.

The documentation for this struct was generated from the following file:

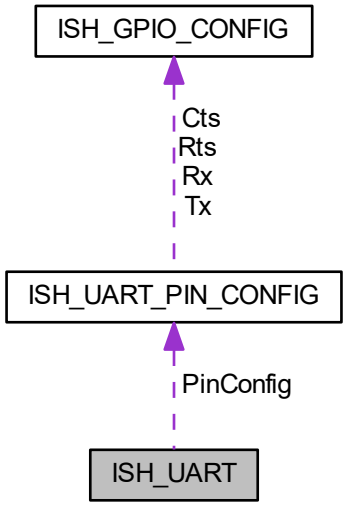
- IshConfig.h

12.74 ISH_UART Struct Reference

Struct contains GPIO pins assigned and signal settings of UART.

```
#include <IshConfig.h>
```

Collaboration diagram for ISH_UART:



Public Attributes

- UINT32 Enable: 1

  ISH UART GPIO pins assigned: ***0: False*** *1: True.*
- UINT32 RsvdBits0: 31

  Reserved Bits.

12.74.1 Detailed Description

Struct contains GPIO pins assigned and signal settings of UART.

Definition at line 128 of file IshConfig.h.

The documentation for this struct was generated from the following file:

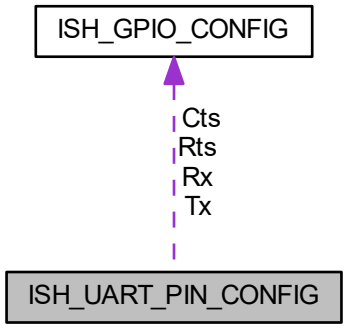
- IshConfig.h

12.75 ISH_UART_PIN_CONFIG Struct Reference

UART signals settings.

`#include <IshConfig.h>`

Collaboration diagram for ISH_UART_PIN_CONFIG:



Public Attributes

- ISH_GPIO_CONFIG Rx
  RXD Pin configuration.
- ISH_GPIO_CONFIG Tx
  TXD Pin configuration.
- ISH_GPIO_CONFIG Rts
  RTS Pin configuration.
- ISH_GPIO_CONFIG Cts
  CTS Pin configuration.

12.75.1 Detailed Description

UART signals settings.

Definition at line 97 of file IshConfig.h.

The documentation for this struct was generated from the following file:

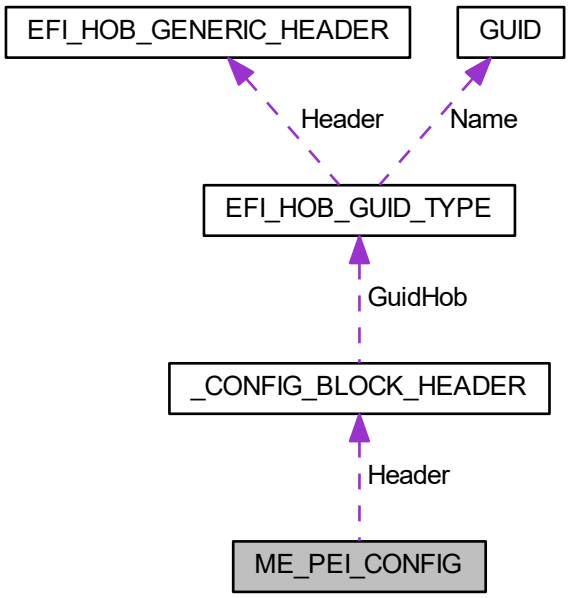
- IshConfig.h

12.76 ME_PEI_CONFIG Struct Reference

ME Pei Post-Memory Configuration Structure.

`#include <MePeiConfig.h>`

Collaboration diagram for ME_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT32 EndOfPostMessage: 2
  0: Disabled; 1: Send in PEI; ***2: Send in DXE*** *- Send EOP at specific phase.*
- UINT32 DisableD0I3SettingForHeci: 1

*(Test) **0: Disable**; 1: Enable - Enable/Disable D0i3 for HECI.*

- UINT32 MeUnconfigOnRtcClear: 2

  Enable/Disable Me Unconfig On Rtc Clear.

- UINT32 MctpBroadcastCycle: 1

  *(Test) **0: Disable**; 1: Enable - Program registers for MCTP Cycle.*

- UINT32 EnforceEDebugMode: 1

  ***0: Disable**; 1: Enable - Enforces ME to enter Enhanced Debug Mode*

- UINT32 CseDataResilience: 1

  *0: Disable; **1: Enable** - CSE data resilience support*

- UINT32 RsvdBits: 24

  Reserved for future use & Config block alignment.

12.76.1 Detailed Description

ME Pei Post-Memory Configuration Structure.

Revision 1:

- Initial version.

Definition at line 104 of file MePeiConfig.h.

12.76.2 Member Data Documentation

12.76.2.1 MeUnconfigOnRtcClear

`UINT32 ME_PEI_CONFIG::MeUnconfigOnRtcClear`

Enable/Disable Me Unconfig On Rtc Clear.

If enabled, BIOS will send MeUnconfigOnRtcClearDisable Msg with parameter 0. It will cause ME to unconfig if RTC is cleared.

- 0: Disable
- **1: Enable**
- 2: Cmos is clear, status unkonwn
- 3: Reserved

Definition at line 117 of file MePeiConfig.h.

The documentation for this struct was generated from the following file:

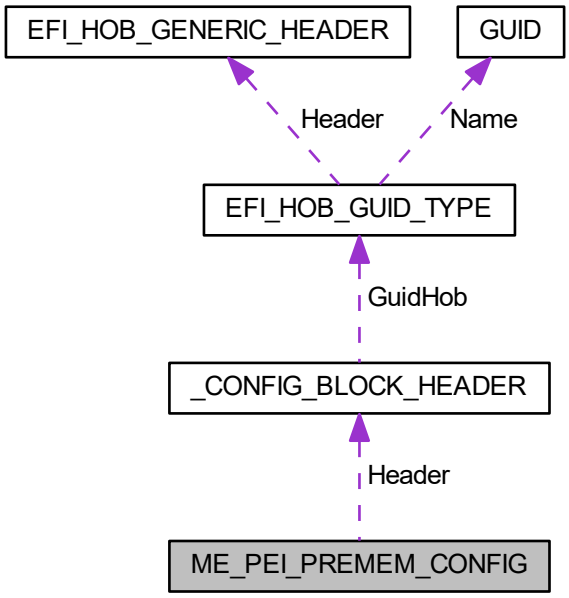
- MePeiConfig.h

12.77 ME_PEI_PREMEM_CONFIG Struct Reference

ME Pei Pre-Memory Configuration Structure.

`#include <MePeiConfig.h>`

Collaboration diagram for ME_PEI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 HeciTimeouts: 1

  *0: Disable; **1: Enable** - HECI Send/Receive Timeouts.*
- UINT32 DidInitStat: 2

  ***(Test) 0: Disabled** 1: ME DID init stat 0 - Success 2: ME DID init stat 1 - No Memory in Channels 3: ME DID init stat 2 - Memory Init Error*
- UINT32 DisableCpuReplacedPolling: 1

  ***(Test) 0: Set to 0 to enable polling for CPU replacement** 1: Set to 1 will disable polling for CPU replacement*
- UINT32 DisableMessageCheck: 1

  ***(Test) 0: ME BIOS will check each messages before sending** 1: ME BIOS always sends messages without checking*
- UINT32 SkipMbpHob: 1

  ***(Test)** The SkipMbpHob policy determines whether ME BIOS Payload data will be requested during boot.*
- UINT32 HeciCommunication2: 1

  ***(Test) 0: Disable**; 1: Enable - Enable/Disable HECI2.*
- UINT32 KtDeviceEnable: 2

*(Test) 0: **POR**; 1: Enable; 2: Disable - Enable/Disable KT Device.*

- UINT32 SkipCpuReplacementCheck: 1

  *(Test) 0: **Disable**; 1: Enable - Enable/Disable to skip CPU replacement check.*

- UINT32 HeciFullTrace: 1

  *(Test) 0: Disable; **1: Enable** - Enable/Disable Full HECI communication information logged to serial*

- UINT32 RsvdBits: 21

  Reserved for future use & Config block alignment.

12.77.1 Detailed Description

ME Pei Pre-Memory Configuration Structure.

Revision 1:

- Initial version.

Definition at line 52 of file MePeiConfig.h.

12.77.2 Member Data Documentation

12.77.2.1 SkipMbpHob

```
UINT32 ME_PEI_PREMEM_CONFIG::SkipMbpHob
```

(Test) The SkipMbpHob policy determines whether ME BIOS Payload data will be requested during boot.

If set to 1, BIOS will not send Get MBP message and not create MBP Hob. **0: ME BIOS will send Get MBP message and create HOB for MBP data** 1: ME BIOS will not send Get MBP message

Definition at line 82 of file MePeiConfig.h.

The documentation for this struct was generated from the following file:

- MePeiConfig.h

12.78 MUX_GPIO_CONFIG Struct Reference

I3C GPIO settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 PadTermination

  GPIO signals pin muxing settings.

12.78.1 Detailed Description

I3C GPIO settings.

Definition at line 261 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

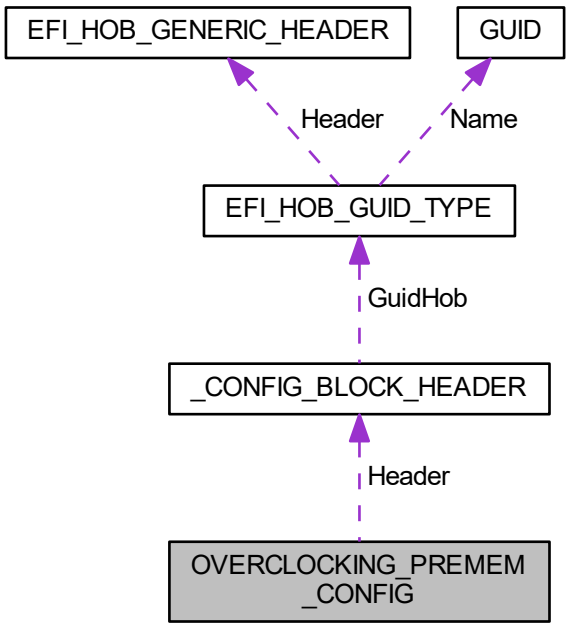
- SerialIoDevices.h

12.79 OVERCLOCKING_PREMEM_CONFIG Struct Reference

Overclocking Configuration Structure.

`#include <OverclockingConfig.h>`

Collaboration diagram for OVERCLOCKING_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 OcSupport: 1

  Overclocking support.
- UINT32 CoreVoltageMode: 1

Core voltage mode, specifies which voltage mode the processor will be operating.

- UINT32 CoreVfPointOffsetMode: 1

  Selects Core Voltage & Frequency Offset between Legacy and Selection modes.

- UINT32 CoreVfConfigScope: 1

  Core VF Configuration Scope.

- UINT32 PerCoreRatioOverride: 1

  Enable or disable Per Core PState OC supported by writing OCMB 0x1D to program new favored core ratio to each Core. ***0: Disable****, 1: enable.*

- UINT32 CoreRatioExtensionMode:1

  Enable or disable Core Ratio above 85 Extension Mode by writing BIOS MB 0x37 to enable FULL_RANGE_MULT↩ IPLIER_UNLOCK_EN. ***0: Disable****, 1: enable.*

- UINT32 RingDownBin: 1

  Ring Downbin enable/disable.

- UINT32 RingVoltageMode: 1

  Ring voltage mode, specifies which voltage mode the processor will be operating.

- UINT32 GtVoltageMode: 1

  Specifies whether GT voltage is operating in Adaptive or Override mode: ***0=Adaptive****, 1=Override.*

- UINT32 SaVoltageMode: 1

  SA/Uncore voltage mode, specifies which voltage mode the processor will be operating.

- UINT32 TvbRatioClipping: 1

  This service controls Core frequency reduction caused by high package temperatures for processors that implement the Intel Thermal Velocity Boost (TVB) feature.

- UINT32 TvbVoltageOptimization: 1

  This service controls thermal based voltage optimizations for processors that implement the Intel Thermal Velocity Boost (TVB) feature.

- UINT32 IaIccUnlimitedMode: 1

  Support IA Unlimited ICCMAX up to maximum value 512A., ***0: Disable****. 1: Enable.*

- UINT32 GtIccUnlimitedMode: 1

  Support GT Unlimited ICCMAX up to maximum value 512A., ***0: Disable****. 1: Enable.*

- UINT32 VccsaBootVoltageSel: 1

  This service controls VCCSA Boot Voltage by programming the EPOC2 bits (strap).

- UINT32 CpuBandgapRefMode: 1

  This service controls CPU Bandgap bypass or not by programming the CPU EPOC2 bits (strap).

- UINT32 VcciaBootVoltageSel: 1

  This service controls VCCIA boot voltage by programming SOC-North soft straps.

- UINT32 BclkAdaptiveVoltage: 1

  Bclk Adaptive Voltage enable/disable. ***0: Disabled****, 1: Enabled. When enabled, the CPU V/F curves are aware of BCLK frequency when calculated.*

- UINT32 RealtimeMemoryTiming: 1

  Enable/Disable the message sent to the CPU to allow realtime memory timing changes after MRC_DONE. ***0=Disable****, 1=Enable.*

- UINT32 Avx2RatioOffset: 5

  AVX2 Ratio Offset. ***0: No offset****. Range is 0-31. Used to lower the AVX ratio to maximize possible ratio for SSE workload.*

- UINT32 Avx3RatioOffset: 5

  Deprecated from Revision 10.

- UINT32 GranularRatioOverride: 1

  Enable or disable OC Granular Ratio Override. ***0: Disable****, 1: enable.*

- UINT32 MemSSVoltageMode: 1

  Memory Subsystem voltage mode, specifies which voltage mode the processor will be operating.

- UINT8 Avx2VoltageScaleFactor

Avx2 Voltage Guardband Scale Factor This controls the AVX2 Voltage Guardband Scale factor applied to AVX2 workloads.

- UINT8 Avx512VoltageScaleFactor

  Avx512 Voltage Guardband Scale Factor This controls the AVX512 Voltage Guardband Scale factor applied to AV↩ X512 workloads.
- UINT8 CoreMaxOcRatio

  Maximum core turbo ratio override allows to increase CPU core frequency beyond the fused max turbo ratio limit (P0).
- UINT8 GtMaxOcRatio

  Maximum GT turbo ratio override: 0=Minimal, 60=Maximum, ***0=AUTO***
- UINT8 RingMaxOcRatio

  Maximum ring ratio override allows to increase CPU ring frequency beyond the fused max ring ratio limit.
- UINT8 TjMaxOffset

  TjMax Offset.
- UINT16 CoreVoltageOverride

  The core voltage override which is applied to the entire range of cpu core frequencies.
- UINT16 CoreAdaptiveVoltage

  Adaptive voltage target used to define the interpolation voltage point when the cpu is operating in adaptive mode range.
- INT16 CoreVoltageOffset

  The core voltage offset applied on top of all other voltage modes.
- UINT16 RingVoltageOverride

  The ring voltage override which is applied to the entire range of cpu ring frequencies.
- UINT16 RingAdaptiveVoltage

  Adaptive voltage target used to define the interpolation voltage point when the ring is operating in adaptive mode range.
- INT16 RingVoltageOffset

  The ring voltage offset applied on top of all other voltage modes.
- UINT16 SaVoltageOverride

  The SA/Uncore voltage override which is applied to the entire range of uncore frequencies.
- UINT16 SaAdaptiveVoltage

  Adaptive voltage target used to define the interpolation voltage point when the SA/Uncore is operating in adaptive mode range.
- UINT16 IaIccMax

  Voltage Regulator Current Limit (Icc Max) for IA, GT and SA.
- INT16 GtVoltageOffset

  The voltage offset applied to GT slice. Valid range from -1000mv to 1000mv: ***0=Minimal****, 1000=Maximum.*
- UINT16 GtVoltageOverride

  The GT voltage override which is applied to the entire range of GT frequencies ***0=Default***
- UINT16 GtAdaptiveVoltage

  The adaptive voltage applied during turbo frequencies. Valid range from 0 to 2000mV: ***0=Minimal****, 2000=Maximum.*
- UINT8 GtVfPointOffsetMode

  Selects GT Voltage & Frequency Point Offset between Legacy and Selection modes.
- UINT8 GtVfPointCount

  Number of supported GR Voltage & Frequency Point.
- INT16 GtVfPointOffset [CPU_OC_MAX_VF_POINTS]

  Array used to specifies the GT Voltage Offset applied to the each selected VF Point.
- UINT8 GtVfPointRatio [CPU_OC_MAX_VF_POINTS]

  Array for the each selected VF Point to display the GT Ration.
- INT16 SaVoltageOffset

  The voltage offset applied to the SA. Valid range from -1000mv to 1000mv: ***0=Default***
- UINT32 CorePllVoltageOffset: 4

  Core PLL voltage offset. ***0: No offset****. Range 0-15 in 17.5mv units.*

- UINT32 RingPllVoltageOffset: 4

  *Ring PLL voltage offset. **0: No offset**. Range 0-15 in 17.5mv units.*
- UINT32 SaPllVoltageOffset: 4

  *SOC System Agent PLL voltage offset. **0: No offset**. Range 0-15 in 17.5mv units.*
- UINT32 IaAtomPllVoltageOffset: 4

  *IA Atom PLL voltage offset. **0: No offset**. Range 0-15 in 17.5mv units.*
- UINT32 McPllVoltageOffset: 4

  *Memory Controller PLL voltage offset. **0: No offset**. Range 0-15 in 17.5mv units.*
- UINT32 CpuSaPllVoltageOffset: 4

  *Cpu SA PLL voltage offset. **0: No offset**. Range 0-15 in 17.5mv units.*
- UINT32 CpuDlvrMode: 1

  *Select Core(s) and RING DLVR Mode, **0: Regulation Mode**. 1: Power Gate Mode.*
- UINT32 SaPllFreqOverride: 1

  *Sa PLL Frequecy Override. **0: 2400MHz**. 1: 1600MHz.*
- UINT32 TscDisableHwFixup: 1

  *Core HW Fixup disable during TSC copy from PMA to APIC, **0: Enable**. 1: Disable.*
- UINT32 SaIccUnlimitedMode: 1

  *Support SA Unlimited ICCMAX up to maximum value 512A., **0: Disable**. 1: Enable.*
- UINT32 RsvdBits1: 4

  Reserved bits.
- UINT64 DisablePerCoreMask

  Core disable mask is a bitwise indication of which core should be disabled, while ActiveCoreCount is to set the number of active cores.
- UINT16 PerCoreHtDisable

  Defines the per-core HT disable mask where: 1 - Disable selected logical core HT, 0 - is ignored.
- UINT8 CoreVfPointCount

  Number of supported Core Voltage & Frequency Point.
- UINT8 CoreVfPointRatio [CPU_OC_MAX_VF_POINTS]

  Array for the each selected VF Point to display the Core Ration. It's only for read command and not for write.
- INT16 CoreVfPointOffset [CPU_OC_MAX_VF_POINTS]

  Array used to specifies the Core Voltage Offset applied to the each selected VF Point.
- UINT8 RingVfPointOffsetMode

  Selects Ring Voltage & Frequency Point Offset between Legacy and Selection modes.
- UINT8 RingVfPointCount

  Number of supported Ring Voltage & Frequency Point.
- INT16 RingVfPointOffset [CPU_OC_MAX_VF_POINTS]

  Array used to specifies the Ring Voltage Offset applied to the each selected VF Point.
- UINT8 RingVfPointRatio [CPU_OC_MAX_VF_POINTS]

  Array for the each selected VF Point to display the Ring Ration. It's only for read command and not for write.
- UINT8 MemSSVfPointCount

  Number of supported Memory Subsystem Voltage & Frequency Point.
- INT16 MemSSVfPointOffset [CPU_OC_MAX_VF_POINTS]

  Array used to specifies the Memory Subsystem Voltage Offset applied to the each selected VF Point.
- UINT8 MemSSVfPointRatio [CPU_OC_MAX_VF_POINTS]

  Array for the each selected VF Point to display the Memory Subsystem Ration. It's only for read command and not for write.
- UINT8 FllOverclockMode

  FLL_OVERCLOCK_MODE: 0x0 = no overclocking, 0x1 = ratio overclocking with nominal (0.5-1x) reference clock frequency, 0x2 = BCLK overclocking with elevated (1-3x) reference clock frequency, 0x3 = BCLK overclocking with extreme elevated (3-5x) reference clock frequency and ratio limited to 63.
- UINT8 CpuD2dRatio

Set CPU D2D Ratio from Range 15 to 40.

- UINT8 UnderVoltProtection

  When UnderVolt Protection is enabled, user will be not be able to program under voltage in OS runtime.
- INT16 PerCoreVoltageOffset [CPU_MAX_BIG_CORES]

  Array used to specifies the selected Core Offset Voltage.
- UINT8 PerCoreVoltageMode [CPU_MAX_BIG_CORES]

  Array used to specifies the selected Core Voltage Mode.
- UINT16 PerCoreVoltageOverride [CPU_MAX_BIG_CORES]

  Array used to specifies the selected Core Voltage Override.
- UINT16 PerCoreAdaptiveVoltage [CPU_MAX_BIG_CORES]

  Array used to specifies the selected Core Adaptive Voltage.
- UINT8 PerAtomClusterVoltageMode [CPU_MAX_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Core Voltage Mode.
- UINT16 PerAtomClusterVoltageOverride [CPU_MAX_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Core Voltage Override.
- UINT16 PerAtomClusterAdaptiveVoltage [CPU_MAX_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Core Adaptive Voltage.
- UINT8 AtomClusterRatio [CPU_MAX_ATOM_CLUSTERS]

  The Atom cluster max ratio overrides.
- UINT32 CpuBclkOcFrequency

  *CPU BCLK OC Frequency in 10KHz units increasing. Value 9800 (10KHz) = 98MHz **0 - Auto**. Range is 8000-50000 (10KHz)*
- UINT32 SocBclkOcFrequency

  *SOC BCLK OC Frequency in 10KHz units increasing. Value 9800 (10KHz) = 98MHz **0 - Auto**. Range is 4000-100000 (10KHz)*
- UINT8 PvdRatioThreshold [2]

  The Pvd Ratio Threshold for SOC/CPU die is the threshold value for input ratio (P0 to Pn) to select the multiplier so that the output is within the DCO frequency range.
- UINT8 PvdMode [2]

  PVD Mode selects Static PVD ratio for SOC/CPU (when PVD THRESHOLD value is 0).
- UINT8 PerCoreGranularityBins [CPU_MAX_BIG_CORES]

  Percore & PerAtomCluster Granularity Bins Override.
- UINT16 MemSSAdaptiveVoltage

  Adaptive voltage target used to define the interpolation voltage point when the cpu is operating in adaptive mode range.
- UINT16 MemSSVoltageOverride

  The Memory Subsystem voltage override which is applied to the entire range of uncore frequencies.
- UINT8 MemSSVfPointOffsetMode

  Selects Memory Subsystem Voltage & Frequency Point Offset between Legacy and Selection modes.
- UINT8 MemSSMaxOcRatio

  Maximum Memory Subsystem ratio override allows to increase memory frequency beyond the fused max ring ratio limit.
- INT16 MemSSVoltageOffset

  The Memory Subsystem voltage offset applied on top of all other voltage modes.
- UINT8 OcTvb

  OC TVB Enable/Disable.
- UINT8 PcoreTvbTempThreshold0

  Temperature Threshold 0 for all Pcores (in degrees C).
- UINT8 PcoreTvbTempThreshold1

  Temperature Threshold 1 for all Pcores (in degrees C).
- UINT8 EcoreTvbTempThreshold0

  Temperature Threshold 0 for all Ecores (in degrees C).

- UINT8 EcoreTvbTempThreshold1

  Temperature Threshold 1 for all Ecores (in degrees C).
- UINT8 TvbConfigLimitSelect

  Limits Select, 0 - Per CCP (Module), Command Param2 = CCP ID, CCP ID can range from 0-15.
- UINT8 PerPcoreRatioDownBinAboveT0 [CPU_MAX_BIG_CORES]

  Down Bins (delta) for Temperature Threshold 0.
- UINT8 PerPcoreRatioDownBinAboveT1 [CPU_MAX_BIG_CORES]

  Down Bins (delta) for Temperature Threshold 1.
- UINT8 PerEcoreCcpRatioDownBinAboveT0 [CPU_MAX_ATOM_CLUSTERS]

  Down Bins (delta) for Temperature Threshold 0.
- UINT8 PerEcoreCcpRatioDownBinAboveT1 [CPU_MAX_ATOM_CLUSTERS]

  Down Bins (delta) for Temperature Threshold 1.
- UINT8 PerPcoreGrRatioDownBinAboveT0 [CPU_MAX_BIG_CORE_TRL_GROUPS]

  Down Bins (delta) for Temperature Threshold 0.
- UINT8 PerPcoreGrRatioDownBinAboveT1 [CPU_MAX_BIG_CORE_TRL_GROUPS]

  Down Bins (delta) for Temperature Threshold 1.
- UINT8 RingTurboThermalProtection

  Ring Turbo Thermal Protection (TTP) control which allows user to disable Turbo Thermal Protection for Ring.
- UINT8 NclkTurboThermalProtection

  NCLK Turbo Thermal Protection (TTP) control which allows user to disable Turbo Thermal Protection for SA.
- UINT8 ComputeDieSscEnable

  Enable or Disable Compute Die SSC configuration.
- UINT8 SocDieSscEnable

  Enable or Disable Soc Die SSC configuration.
- UINT8 PerCoreMaxRatio [CPU_MAX_BIG_CORES]

  Array used to specifies the selected Core Max Ratio.
- UINT8 PerAtomClusterMaxRatio [CPU_MAX_ATOM_CLUSTERS]

  Array used to specifies the selected Atom Cluster Max Ratio.
- UINT8 NguMaxOcRatio

  Maximum NGU ratio override allows to increase CPU NGU frequency beyond the fused max NGU ratio limit.
- UINT8 NguRatio

  Sets the Ratio for NGU using SAVG B2P Mailbox cmd 0x22 and subcommand 0x1.
- UINT16 NguVoltageOverride

  The NGU voltage override which is applied to the entire range of cpu NGU frequencies.
- UINT16 NguAdaptiveVoltage

  The adaptive voltage applied during turbo frequencies. Valid range from 0 to 2000mV: ***0=Minimal****, 2000=Maximum.*
- INT16 NguVoltageOffset

  The NGU voltage offset applied on top of all other voltage modes.
- UINT8 NguVfPointOffsetMode

  Selects NGU Voltage & Frequency Point Offset between Legacy and Selection modes.
- UINT8 NguVfPointCount

  Number of supported NGU Voltage & Frequency Point.
- INT16 NguVfPointOffset [CPU_OC_MAX_VF_POINTS]

  Array used to specifies the NGU Voltage Offset applied to the each selected VF Point.
- UINT8 NguVfPointRatio [CPU_OC_MAX_VF_POINTS]

  Array for the each selected VF Point to display the NGU Ration. It's only for read command and not for write.
- UINT8 ProcessVmaxLimit

  Disabling Process Vmax Limit will allow user to set any voltage.
- UINT8 PcorePowerDensityThrottle

  Pcore Power Density Throttle control allows user to disable P-core throttling for power density protection.
- UINT16 MaxVoltageLimit [MAX_OC_DOMAINS]

*Array for the each Domain to set OC Max Voltage limits **Default: 0** Range: 0 to 2000.*

- UINT8 CorePllCurrentRefTuningOffset

  *Core PLL Current Reference Tuning Offset. **0: No offset**. Range 0-15.*

- UINT8 RingPllCurrentRefTuningOffset

  Deprecated from Revision 11.

- UINT8 IaAtomPllCurrentRefTuningOffset

  *IaAtom PLL Current Reference Tuning Offset. **0: No offset**. Range 0-15.*

- UINT8 CoreMinRatio

  Request Core Min Ratio.

- UINT8 PerCoreDisableConfiguration

  This configuration allows OC user to either use legacy num of cores option or the OC specific per core disable configuration.

- UINT8 NpuMaxOcRatio

  Deprecated from Revision 13.

- UINT8 NpuVoltageMode

  Deprecated from Revision 13.

- UINT16 NpuVoltageOverride

  Deprecated from Revision 13.

- UINT16 NpuAdaptiveVoltage

  Deprecated from Revision 13.

- INT16 NpuVoltageOffset

  Deprecated from Revision 13.

- UINT8 ForcePcoreResidency

  *Enable or disable Force Pcore Residency. **0: Disable**, 1: enable.*

- UINT8 VrLimitBypass

  *Enable or disable VR Limit Bypass. **0: Disable**, 1: enable.*

- UINT8 TurboThermalProtection

  Turbo Thermal Protection (TTP) control allows user to disable Turbo Thermal Protection.

12.79.1 Detailed Description

Overclocking Configuration Structure.

Revision 1: - Initial version. **Revision 2**: - Added ProcessVmaxLimit **Revision 3**: - Added PcorePowerDensity↩ Throttle **Revision 4**: - Added MaxVoltageLimit **Revision 5**: - Added CorePllCurrentRefTuningOffset,RingPll↩ CurrentRefTuningOffset,IaAtomPllCurrentRefTuningOffset **Revision 6**: - Added CoreMinRatio **Revision 7**: - Added PerCoreDisableConfiguration **Revision 8**: - Added NguRatio **Revision 9**: - Added NpuMaxOcRatio,NpuVoltage↩ Mode,NpuVoltageOverride,NpuAdaptiveVoltage,NpuVoltageOffset **Revision 10**: - Deprecated Avx3RatioOffset and Avx512VoltageScaleFactor **Revision 11**: - Deprecated RingPllCurrentRefTuningOffset **Revision 12**: - Added ForcePcoreResidency and VrLimitBypass **Revision 13**: - Deprecated NpuMaxOcRatio,NpuVoltageMode,Npu↩ VoltageOverride,NpuAdaptiveVoltage,NpuVoltageOffset **Revision 14**: - Added TurboThermalProtection **Revision 15**: - Added RingTurboThermalProtectionand,NclkTurboThermalProtection

Definition at line 73 of file OverclockingConfig.h.

12.79.2 Member Data Documentation

12.79.2.1 AtomClusterRatio

`UINT8 OVERCLOCKING_PREMEM_CONFIG::AtomClusterRatio[CPU_MAX_ATOM_CLUSTERS]`

The Atom cluster max ratio overrides.

Default: 0 Range: 0 to 120.

Definition at line 370 of file OverclockingConfig.h.

12.79.2.2 Avx2VoltageScaleFactor

`UINT8 OVERCLOCKING_PREMEM_CONFIG::Avx2VoltageScaleFactor`

Avx2 Voltage Guardband Scale Factor This controls the AVX2 Voltage Guardband Scale factor applied to AVX2 workloads.

Valid range is 0-200 in 1/100 units, where a value of 125 would apply a 1.25 scale factor. A value of 0 means no scale factor applied (no change to voltage on AVX commands) A value of 100 applies the default voltage guardband values (1.0 factor). A value $>$ 100 will increase the voltage guardband on AVX2 workloads. A value $<$ 100 will decrease the voltage guardband on AVX2 workloads. **0. No scale factor applied**

Definition at line 183 of file OverclockingConfig.h.

12.79.2.3 Avx512VoltageScaleFactor

`UINT8 OVERCLOCKING_PREMEM_CONFIG::Avx512VoltageScaleFactor`

Avx512 Voltage Guardband Scale Factor This controls the AVX512 Voltage Guardband Scale factor applied to AVX512 workloads.

Valid range is 0-200 in 1/100 units, where a value of 125 would apply a 1.25 scale factor. A value of 0 means no scale factor applied (no change to voltage on AVX commands) A value of 100 applies the default voltage guardband values (1.0 factor). A value $>$ 100 will increase the voltage guardband on AVX512 workloads. A value $<$ 100 will decrease the voltage guardband on AVX512 workloads. **0. No scale factor applied**

Definition at line 194 of file OverclockingConfig.h.

12.79.2.4 ComputeDieSscEnable

`UINT8 OVERCLOCKING_PREMEM_CONFIG::ComputeDieSscEnable`

Enable or Disable Compute Die SSC configuration.

- **0: Disable**
- 1: Enable

Definition at line 519 of file OverclockingConfig.h.

12.79.2.5 CoreAdaptiveVoltage

`UINT16 OVERCLOCKING_PREMEM_CONFIG::CoreAdaptiveVoltage`

Adaptive voltage target used to define the interpolation voltage point when the cpu is operating in adaptive mode range.

Used when CoreVoltageMode = Adaptive. **0. no override**. Range 0-2000mV.

Definition at line 223 of file OverclockingConfig.h.

12.79.2.6 CoreMaxOcRatio

`UINT8 OVERCLOCKING_PREMEM_CONFIG::CoreMaxOcRatio`

Maximum core turbo ratio override allows to increase CPU core frequency beyond the fused max turbo ratio limit (P0).

0. no override/HW defaults.. Range non-turbo max - 120 (Previously max turbo is 83).

Definition at line 200 of file OverclockingConfig.h.

12.79.2.7 CoreMinRatio

`UINT8 OVERCLOCKING_PREMEM_CONFIG::CoreMinRatio`

Request Core Min Ratio.

Limit Core minimum ratio for extreme overclocking. Default 0 indicates no request.

Definition at line 592 of file OverclockingConfig.h.

12.79.2.8 CoreVfConfigScope

`UINT32 OVERCLOCKING_PREMEM_CONFIG::CoreVfConfigScope`

Core VF Configuration Scope.

Alows both all-core VF curve or per-core VF curve configuration. **0: All-core Scope, setting the VF curve for all cores.** 1: Per-core Scope, setting the VF curve per-core.

Definition at line 102 of file OverclockingConfig.h.

12.79.2.9 CoreVfPointOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::CoreVfPointOffset[CPU_OC_MAX_VF_POINTS]
```

Array used to specifies the Core Voltage Offset applied to the each selected VF Point.

This voltage is specified in millivolts. **Default: 0** Range: 0 to 1000

Definition at line 316 of file OverclockingConfig.h.

12.79.2.10 CoreVfPointOffsetMode

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::CoreVfPointOffsetMode
```

Selects Core Voltage & Frequency Offset between Legacy and Selection modes.

Need Reset System after enabling OverClocking Feature to initialize the default value. **0: In Legacy Mode, setting a global offset for the entire VF curve.** 1: In Selection Mode, setting a selected VF point.

Definition at line 95 of file OverclockingConfig.h.

12.79.2.11 CoreVoltageMode

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::CoreVoltageMode
```

Core voltage mode, specifies which voltage mode the processor will be operating.

0: Adaptive Mode allows the processor to interpolate a voltage curve when beyond fused P0 range; 1: Override, sets one voltage for the entire frequency range, Pn-P0.

Definition at line 88 of file OverclockingConfig.h.

12.79.2.12 CoreVoltageOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::CoreVoltageOffset
```

The core voltage offset applied on top of all other voltage modes.

This offset is applied over the entire frequency range. This is a 2's complement number in mV units. **Default: 0** Range: -1000 to 1000.

Definition at line 228 of file OverclockingConfig.h.

12.79.2.13 CoreVoltageOverride

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::CoreVoltageOverride
```

The core voltage override which is applied to the entire range of cpu core frequencies.

Used when CoreVoltageMode = Override. **0. no override**. Range 0-2000 mV.

Definition at line 217 of file OverclockingConfig.h.

12.79.2.14 CpuBandgapRefMode

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::CpuBandgapRefMode
```

This service controls CPU Bandgap bypass or not by programming the CPU EPOC2 bits (strap).

Default: 0: Normal 1: Bandgap Bypassed

Definition at line 150 of file OverclockingConfig.h.

12.79.2.15 CpuD2dRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::CpuD2dRatio
```

Set CPU D2D Ratio from Range 15 to 40.

Default 0 indicates no setting.

Definition at line 348 of file OverclockingConfig.h.

12.79.2.16 DisablePerCoreMask

```
UINT64 OVERCLOCKING_PREMEM_CONFIG::DisablePerCoreMask
```

Core disable mask is a bitwise indication of which core should be disabled, while ActiveCoreCount is to set the number of active cores.

Each bit represents physical core id for both P-core and E-core 1 - Disable 0 - Ignored Default is 0, means no Core disable mask setting.

Definition at line 302 of file OverclockingConfig.h.

12.79.2.17 EcoreTvbTempThreshold0

`UINT8 OVERCLOCKING_PREMEM_CONFIG::EcoreTvbTempThreshold0`

Temperature Threshold 0 for all Ecores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T0 from the resolved OC Ratio, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **70** . Range 0-100.

Definition at line 451 of file OverclockingConfig.h.

12.79.2.18 EcoreTvbTempThreshold1

`UINT8 OVERCLOCKING_PREMEM_CONFIG::EcoreTvbTempThreshold1`

Temperature Threshold 1 for all Ecores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T1 from the resolved OC Ratio, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **100** . Range 0-100.

Definition at line 457 of file OverclockingConfig.h.

12.79.2.19 GtVfPointOffset

`INT16 OVERCLOCKING_PREMEM_CONFIG::GtVfPointOffset[CPU_OC_MAX_VF_POINTS]`

Array used to specifies the GT Voltage Offset applied to the each selected VF Point.

This voltage is specified in millivolts. **Default: 0** Range: 0 to 1000

Definition at line 280 of file OverclockingConfig.h.

12.79.2.20 GtVfPointOffsetMode

`UINT8 OVERCLOCKING_PREMEM_CONFIG::GtVfPointOffsetMode`

Selects GT Voltage & Frequency Point Offset between Legacy and Selection modes.

Need Reset System after enabling OverClocking Feature to initialize the default value. **0: In Legacy Mode, setting a global offset for the entire VF curve.** 1: In Selection modes, setting a selected VF point.

Definition at line 274 of file OverclockingConfig.h.

12.79.2.21 IaIccMax

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::IaIccMax
```

Voltage Regulator Current Limit (Icc Max) for IA, GT and SA.

This value represents the Maximum instantaneous current allowed at any given time. The value is represented in 1/4 A increments. A value of 400 = 100A. **0 (HW default)**. Range is 4-2047.

Definition at line 262 of file OverclockingConfig.h.

12.79.2.22 MemSSAdaptiveVoltage

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::MemSSAdaptiveVoltage
```

Adaptive voltage target used to define the interpolation voltage point when the cpu is operating in adaptive mode range.

Used when MemSSVoltageMode = Adaptive. **0. no override**. Range 0-2000mV.

Definition at line 401 of file OverclockingConfig.h.

12.79.2.23 MemSSMaxOcRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::MemSSMaxOcRatio
```

Maximum Memory Subsystem ratio override allows to increase memory frequency beyond the fused max ring ratio limit.

0. no override/HW defaults.. Range non-turbo max - 109.

Definition at line 419 of file OverclockingConfig.h.

12.79.2.24 MemSSVfPointOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::MemSSVfPointOffset[CPU_OC_MAX_VF_POINTS]
```

Array used to specifies the Memory Subsystem Voltage Offset applied to the each selected VF Point.

This voltage is specified in millivolts. **Default: 0** Range: 0 to 1000

Definition at line 336 of file OverclockingConfig.h.

12.79.2.25 MemSSVfPointOffsetMode

`UINT8 OVERCLOCKING_PREMEM_CONFIG::MemSSVfPointOffsetMode`

Selects Memory Subsystem Voltage & Frequency Point Offset between Legacy and Selection modes.

Need Reset System after enabling OverClocking Feature to Initialize the default value. **0: In Legacy Mode, setting a global offset for the entire VF curve.** 1: In Selection modes, setting a selected VF point.

Definition at line 414 of file OverclockingConfig.h.

12.79.2.26 MemSSVoltageMode

`UINT32 OVERCLOCKING_PREMEM_CONFIG::MemSSVoltageMode`

Memory Subsystem voltage mode, specifies which voltage mode the processor will be operating.

0: Adaptive Mode allows the processor to interpolate a voltage curve when beyond fused P0 range; 1: Override, sets one voltage for the entire frequency range, Pn-P0.

Definition at line 166 of file OverclockingConfig.h.

12.79.2.27 MemSSVoltageOffset

`INT16 OVERCLOCKING_PREMEM_CONFIG::MemSSVoltageOffset`

The Memory Subsystem voltage offset applied on top of all other voltage modes.

This offset is applied over the entire frequency range. This is a 2's complement number in mV units. **Default: 0** Range: -1000 to 1000.

Definition at line 424 of file OverclockingConfig.h.

12.79.2.28 MemSSVoltageOverride

`UINT16 OVERCLOCKING_PREMEM_CONFIG::MemSSVoltageOverride`

The Memory Subsystem voltage override which is applied to the entire range of uncore frequencies.

Used when MemSSVoltageMode = Override. **0. no override**. Range 0-2000 mV.

Definition at line 407 of file OverclockingConfig.h.

12.79.2.29 NclkTurboThermalProtection

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::NclkTurboThermalProtection
```

NCLK Turbo Thermal Protection (TTP) control which allows user to disable Turbo Thermal Protection for SA.

This is a sticky bit, Once set, will remain set until cold or warm reset is performed. 0: Disable; **1: Enable**

Definition at line 512 of file OverclockingConfig.h.

12.79.2.30 NguMaxOcRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::NguMaxOcRatio
```

Maximum NGU ratio override allows to increase CPU NGU frequency beyond the fused max NGU ratio limit.

0. no override/HW defaults.. Range 0-34.

Definition at line 536 of file OverclockingConfig.h.

12.79.2.31 NguRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::NguRatio
```

Sets the Ratio for NGU using SAVG B2P Mailbox cmd 0x22 and subcommand 0x1.

When this value is zero, dynamic mode is selected and NGU ratio can be modified using OCMB cmd 0x11. When valid ratio value is set, static mode is selected with the fixed ratio specified by this value.

Definition at line 542 of file OverclockingConfig.h.

12.79.2.32 NguVfPointOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::NguVfPointOffset[CPU_OC_MAX_VF_POINTS]
```

Array used to specifies the NGU Voltage Offset applied to the each selected VF Point.

This voltage is specified in millivolts. **Default: 0** Range: 0 to 1000

Definition at line 567 of file OverclockingConfig.h.

12.79.2.33 NguVfPointOffsetMode

`UINT8 OVERCLOCKING_PREMEM_CONFIG::NguVfPointOffsetMode`

Selects NGU Voltage & Frequency Point Offset between Legacy and Selection modes.

Need Reset System after enabling OverClocking Feature to initialize the default value. **0: In Legacy Mode, setting a global offset for the entire VF curve.** 1: In Selection modes, setting a selected VF point.

Definition at line 561 of file OverclockingConfig.h.

12.79.2.34 NguVoltageOffset

`INT16 OVERCLOCKING_PREMEM_CONFIG::NguVoltageOffset`

The NGU voltage offset applied on top of all other voltage modes.

This offset is applied over the entire frequency range. This is a 2's complement number in mV units. **Default: 0** Range: -1000 to 1000.

Definition at line 554 of file OverclockingConfig.h.

12.79.2.35 NguVoltageOverride

`UINT16 OVERCLOCKING_PREMEM_CONFIG::NguVoltageOverride`

The NGU voltage override which is applied to the entire range of cpu NGU frequencies.

Used when NguVoltageMode = Override. **0. no override**. Range 0-2000 mV.

Definition at line 548 of file OverclockingConfig.h.

12.79.2.36 OcSupport

`UINT32 OVERCLOCKING_PREMEM_CONFIG::OcSupport`

Overclocking support.

This controls whether OC mailbox transactions are sent. If disabled, all policies in this config block besides OcSupport and OcLock will be ignored. **0: Disable**; 1: Enable.

Note

If PcdOverclockEnable is disabled, this should also be disabled.

Definition at line 82 of file OverclockingConfig.h.

12.79.2.37 OcTvb

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::OcTvb
```

OC TVB Enable/Disable.

This control will come into picture only when TVB Ratio clipping is enabled from OCMB 0x18/0x19 (bit 2). When disabled, BIOS will not send OC TVB parameter commands 0x24/0x25. When OcTvb is enabled, BIOS will allow user to modify and program new parameters for temperature thresholds T0, T1 and delta DownBins for T0 and T1. **0: Disable**, 1: Enable.

Definition at line 433 of file OverclockingConfig.h.

12.79.2.38 PcorePowerDensityThrottle

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PcorePowerDensityThrottle
```

Pcore Power Density Throttle control allows user to disable P-core throttling for power density protection.

This is a sticky bit, Once set, will remain set until cold or warm reset is performed. MailBox control is to disable the Pcore Power Density Throttling (reverse encoding is used). 0: Disable;**1: Enable**;

Definition at line 582 of file OverclockingConfig.h.

12.79.2.39 PcoreTvbTempThreshold0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PcoreTvbTempThreshold0
```

Temperature Threshold 0 for all Pcores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T0 from the resolved OC Ratio, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **70** . Range 0-100.

Definition at line 439 of file OverclockingConfig.h.

12.79.2.40 PcoreTvbTempThreshold1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PcoreTvbTempThreshold1
```

Temperature Threshold 1 for all Pcores (in degrees C).

Running ABOVE this temperature will clip delta Down Bins for T1 from the resolved OC Ratio, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **100** . Range 0-100.

Definition at line 445 of file OverclockingConfig.h.

12.79.2.41 PerCoreDisableConfiguration

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerCoreDisableConfiguration
```

This configuration allows OC user to either use legacy num of cores option or the OC specific per core disable configuration.

0: Disable, 1: Enable

Definition at line 596 of file OverclockingConfig.h.

12.79.2.42 PerCoreGranularityBins

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerCoreGranularityBins[CPU_MAX_BIG_CORES]
```

Percore & PerAtomCluster Granularity Bins Override.

Array used to specifies the selected Core & AtomCluster Granularity Bins. It's a negative offset to current resolved P0 ratio, programming Granularity Bins will always reduce the core frequency. value is how many bins of 'Granularity Units' should be subtracted from the otherwise-configured P0 frequency. **value = 0 (default)** Range 0-255.

Definition at line 394 of file OverclockingConfig.h.

12.79.2.43 PerCoreHtDisable

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::PerCoreHtDisable
```

Defines the per-core HT disable mask where: 1 - Disable selected logical core HT, 0 - is ignored.

Input is in HEX and each bit maps to a logical core. Ex. A value of '1F' would disable HT for cores 4,3,2,1 and 0. **Default is 0**, all cores have HT enabled. Range is 0 - 0x1FF. You can only disable up to MAX_CORE_COUNT - 1.

Definition at line 308 of file OverclockingConfig.h.

12.79.2.44 PerCoreVoltageOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::PerCoreVoltageOffset[CPU_MAX_BIG_CORES]
```

Array used to specifies the selected Core Offset Voltage.

This voltage is specified in millivolts.

Definition at line 358 of file OverclockingConfig.h.

12.79.2.45 PerEcoreCcpRatioDownBinAboveT0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerEcoreCcpRatioDownBinAboveT0[CPU_MAX_ATOM_CLUSTERS]
```

Down Bins (delta) for Temperature Threshold 0.

When running above T0, the ratio of Per Ecore module will be clipped by MAX_RATIO[n]-this value, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **0** . Range 0-10.

Definition at line 482 of file OverclockingConfig.h.

12.79.2.46 PerEcoreCcpRatioDownBinAboveT1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerEcoreCcpRatioDownBinAboveT1[CPU_MAX_ATOM_CLUSTERS]
```

Down Bins (delta) for Temperature Threshold 1.

When running above T1, the ratio of Per Ecore module will be clipped by MAX_RATIO[n]-this value, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **0** . Range 0-10.

Definition at line 488 of file OverclockingConfig.h.

12.79.2.47 PerPcoreGrRatioDownBinAboveT0

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreGrRatioDownBinAboveT0[CPU_MAX_BIG_CORE_TRL_GROUPS]
```

Down Bins (delta) for Temperature Threshold 0.

When running above T0, the ratio of Per Pcore group will be clipped by MAX_RATIO[n]-this value, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **1** . Range 0-10.

Definition at line 494 of file OverclockingConfig.h.

12.79.2.48 PerPcoreGrRatioDownBinAboveT1

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreGrRatioDownBinAboveT1[CPU_MAX_BIG_CORE_TRL_GROUPS]
```

Down Bins (delta) for Temperature Threshold 1.

When running above T1, the ratio of Per Pcore group will be clipped by MAX_RATIO[n]-this value, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **2** . Range 0-10.

Definition at line 500 of file OverclockingConfig.h.

12.79.2.49 PerPcoreRatioDownBinAboveT0

`UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreRatioDownBinAboveT0[CPU_MAX_BIG_CORES]`

Down Bins (delta) for Temperature Threshold 0.

When running above T0, the ratio of Per Pcore module will be clipped by MAX_RATIO[n]-this value, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **1** . Range 0-10.

Definition at line 470 of file OverclockingConfig.h.

12.79.2.50 PerPcoreRatioDownBinAboveT1

`UINT8 OVERCLOCKING_PREMEM_CONFIG::PerPcoreRatioDownBinAboveT1[CPU_MAX_BIG_CORES]`

Down Bins (delta) for Temperature Threshold 1.

When running above T1, the ratio of Per Pcore module will be clipped by MAX_RATIO[n]-this value, when TVB ratio clipping is enabled from OCMB 0x18 Bit 2 = 0 (Enabled). **2** . Range 0-10.

Definition at line 476 of file OverclockingConfig.h.

12.79.2.51 ProcessVmaxLimit

`UINT8 OVERCLOCKING_PREMEM_CONFIG::ProcessVmaxLimit`

Disabling Process Vmax Limit will allow user to set any voltage.

If this limit is not disabled then, request above factory set limits will be rejected. This is a sticky bit, BIOS can only set this bit, it cannot be cleared by mailbox command. This setting persists over warm reset, and cleared on cold reset. MailBox control is to disable the ProcessVmaxLimit (reverse encoding is used). 0: Disable;**1: Enable**; Note: Disabling the voltage limit checks may cause permanent damage to processor. Note: This bit cannot be set after BIOS_RST_CPL.

Definition at line 576 of file OverclockingConfig.h.

12.79.2.52 PvdMode

`UINT8 OVERCLOCKING_PREMEM_CONFIG::PvdMode[2]`

PVD Mode selects Static PVD ratio for SOC/CPU (when PVD THRESHOLD value is 0).

0x0 = div-1 (VCO = Output clock), 0x1 = div-2 (VCO = 2x Output clock) 0x2 = div-4 (VCO = 4x Output clock), 0x3 = div-8 (VCO = 8x Output clock)

Definition at line 386 of file OverclockingConfig.h.

12.79.2.53 PvdRatioThreshold

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::PvdRatioThreshold[2]
```

The Pvd Ratio Threshold for SOC/CPU die is the threshold value for input ratio (P0 to Pn) to select the multiplier so that the output is within the DCO frequency range.

This threshold is applied to SA and MC/CMI PLL for SOC die and SA, Ring and Atom PLL when CPU die is selected. Range 0-63. When the threshold is 0, static PVD ratio is selected based on the PVD Mode for SOC. **0: Default**.

Definition at line 380 of file OverclockingConfig.h.

12.79.2.54 RingAdaptiveVoltage

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::RingAdaptiveVoltage
```

Adaptive voltage target used to define the interpolation voltage point when the ring is operating in adaptive mode range.

Used when RingVoltageMode = Adaptive. **0. no override**. Range 0-2000mV.

Definition at line 240 of file OverclockingConfig.h.

12.79.2.55 RingDownBin

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::RingDownBin
```

Ring Downbin enable/disable.

When enabled, the CPU will force the ring ratio to be lower than the core ratio. Disabling will allow the ring and core ratios to run at the same frequency. Uses OC Mailbox command 0x19. 0: Disables Ring Downbin feature. **1: Enables Ring downbin feature.**

Definition at line 112 of file OverclockingConfig.h.

12.79.2.56 RingMaxOcRatio

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::RingMaxOcRatio
```

Maximum ring ratio override allows to increase CPU ring frequency beyond the fused max ring ratio limit.

0. no override/HW defaults.. Range non-turbo max - 85.

Definition at line 206 of file OverclockingConfig.h.

12.79.2.57 RingTurboThermalProtection

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::RingTurboThermalProtection
```

Ring Turbo Thermal Protection (TTP) control which allows user to disable Turbo Thermal Protection for Ring.

This is a sticky bit, Once set, will remain set until cold or warm reset is performed. 0: Disable; **1: Enable**

Definition at line 506 of file OverclockingConfig.h.

12.79.2.58 RingVfPointOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::RingVfPointOffset[CPU_OC_MAX_VF_POINTS]
```

Array used to specifies the Ring Voltage Offset applied to the each selected VF Point.

This voltage is specified in millivolts. **Default: 0** Range: 0 to 1000

Definition at line 329 of file OverclockingConfig.h.

12.79.2.59 RingVfPointOffsetMode

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::RingVfPointOffsetMode
```

Selects Ring Voltage & Frequency Point Offset between Legacy and Selection modes.

Need Reset System after enabling OverClocking Feature to initialize the default value. **0: In Legacy Mode, setting a global offset for the entire VF curve.** 1: In Selection modes, setting a selected VF point.

Definition at line 323 of file OverclockingConfig.h.

12.79.2.60 RingVoltageMode

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::RingVoltageMode
```

Ring voltage mode, specifies which voltage mode the processor will be operating.

0: Adaptive Mode allows the processor to interpolate a voltage curve when beyond fused P0 range; 1: Override, sets one voltage for the entire frequency range, Pn-P0.

Definition at line 118 of file OverclockingConfig.h.

12.79.2.61 RingVoltageOffset

```
INT16 OVERCLOCKING_PREMEM_CONFIG::RingVoltageOffset
```

The ring voltage offset applied on top of all other voltage modes.

This offset is applied over the entire frequency range. This is a 2's complement number in mV units. **Default: 0** Range: -1000 to 1000.

Definition at line 245 of file OverclockingConfig.h.

12.79.2.62 RingVoltageOverride

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::RingVoltageOverride
```

The ring voltage override which is applied to the entire range of cpu ring frequencies.

Used when RingVoltageMode = Override. **0. no override**. Range 0-2000 mV.

Definition at line 234 of file OverclockingConfig.h.

12.79.2.63 SaAdaptiveVoltage

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::SaAdaptiveVoltage
```

Adaptive voltage target used to define the interpolation voltage point when the SA/Uncore is operating in adaptive mode range.

Used when SaVoltageMode = Adaptive. **0. no override**. Range 0-2000mV.

Definition at line 257 of file OverclockingConfig.h.

12.79.2.64 SaVoltageMode

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::SaVoltageMode
```

SA/Uncore voltage mode, specifies which voltage mode the processor will be operating.

0: Adaptive Mode allows the processor to interpolate a voltage curve when beyond fused P0 range; 1: Override, sets one voltage for the entire frequency range, Pn-P0.

Definition at line 125 of file OverclockingConfig.h.

12.79.2.65 SaVoltageOverride

```
UINT16 OVERCLOCKING_PREMEM_CONFIG::SaVoltageOverride
```

The SA/Uncore voltage override which is applied to the entire range of uncore frequencies.

Used when SaVoltageMode = Override. **0. no override**. Range 0-2000 mV.

Definition at line 251 of file OverclockingConfig.h.

12.79.2.66 SocDieSscEnable

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::SocDieSscEnable
```

Enable or Disable Soc Die SSC configuration.

- **0: Disable**
- 1: Enable

Definition at line 526 of file OverclockingConfig.h.

12.79.2.67 TjMaxOffset

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::TjMaxOffset
```

TjMax Offset.

Specified value here is clipped by pCode (125 - TjMax Offset) to support TjMax in the range of 62 to 115 deg Celsius. **Default: 0 Hardware Defaults** Range 10 to 63. 0 = No offset / Keep HW default.

Definition at line 211 of file OverclockingConfig.h.

12.79.2.68 TurboThermalProtection

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::TurboThermalProtection
```

Turbo Thermal Protection (TTP) control allows user to disable Turbo Thermal Protection.

This is a sticky bit, Once set, will remain set until cold or warm reset is performed. 0: Disable; **1: Enable**

Definition at line 611 of file OverclockingConfig.h.

12.79.2.69 TvbConfigLimitSelect

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::TvbConfigLimitSelect
```

Limits Select, 0 - Per CCP (Module), Command Param2 = CCP ID, CCP ID can range from 0-15.

1 - Per Pcore group number, Command Param2 = Pcore Group Number (0-7), a Pcore active group has number of active cores as present in MSR 0x1AE. **0** .

Definition at line 464 of file OverclockingConfig.h.

12.79.2.70 TvbRatioClipping

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::TvbRatioClipping
```

This service controls Core frequency reduction caused by high package temperatures for processors that implement the Intel Thermal Velocity Boost (TVB) feature.

It is required to be disabled for supporting overclocking at frequencies higher than the default max turbo frequency. 0: Disables TVB ratio clipping. **1: Enables TVB ratio clipping**.

Definition at line 132 of file OverclockingConfig.h.

12.79.2.71 TvbVoltageOptimization

```
UINT32 OVERCLOCKING_PREMEM_CONFIG::TvbVoltageOptimization
```

This service controls thermal based voltage optimizations for processors that implement the Intel Thermal Velocity Boost (TVB) feature.

0: Disables TVB voltage optimization. **1: Enables TVB voltage optimization.**

Definition at line 138 of file OverclockingConfig.h.

12.79.2.72 UnderVoltProtection

```
UINT8 OVERCLOCKING_PREMEM_CONFIG::UnderVoltProtection
```

When UnderVolt Protection is enabled, user will be not be able to program under voltage in OS runtime.

0: protection is disabled, **1:protection is enabled** .

Definition at line 353 of file OverclockingConfig.h.

12.79.2.73 VcciaBootVoltageSel

`UINT32 OVERCLOCKING_PREMEM_CONFIG::VcciaBootVoltageSel`

This service controls VCCIA boot voltage by programming SOC-North soft straps.

Default: 0: Nominal 1: High Voltage (support VCCIA boot voltage higher than 1.65v (max 2.01v))

Definition at line 155 of file OverclockingConfig.h.

12.79.2.74 VccsaBootVoltageSel

`UINT32 OVERCLOCKING_PREMEM_CONFIG::VccsaBootVoltageSel`

This service controls VCCSA Boot Voltage by programming the EPOC2 bits (strap).

Default: 0: Nominal 1: High Voltage(up to 1.2/1.3V)

Definition at line 145 of file OverclockingConfig.h.

The documentation for this struct was generated from the following file:

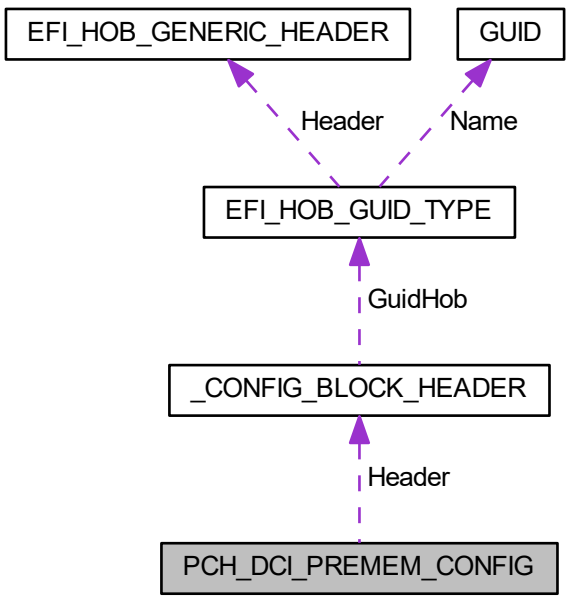
- OverclockingConfig.h

12.80 PCH_DCI_PREMEM_CONFIG Struct Reference

The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI)

`#include <DciConfig.h>`

Collaboration diagram for PCH_DCI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- UINT8 DciEn
  DCI enable.
- UINT8 DciDbcMode
  USB DbC enable mode.
- UINT8 DciUsb3TypecUfpDbg
  USB3 Type-C UFP2DFP kenel / platform debug support.
- UINT8 DciClkEnable
  Determine if to enable or disable DCI clock request in lowest power state.
- UINT8 KeepEarlyTrace
  Early trace is activated by default.

12.80.1 Detailed Description

The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI)

Revision 1:

- Initial version.

Definition at line 74 of file DciConfig.h.

12.80.2 Member Data Documentation

12.80.2.1 DciClkEnable

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciClkEnable
```

Determine if to enable or disable DCI clock request in lowest power state.

0:Disabled; **1:Enabled**

Definition at line 101 of file DciConfig.h.

12.80.2.2 DciDbcMode

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciDbcMode
```

USB DbC enable mode.

Disabled: Clear both USB2/3DBCEN; USB2: Set USB2DBCEN; USB3: Set USB3DBCEN; Both: Set both U↩ SB2/3DBCEN; No Change: Comply with HW value Refer to definition of DCI_USB_DBC_MODE for supported settings. 0:Disabled; 1:USB2; 2:USB3; 3:Both; **4:No Change**

Definition at line 88 of file DciConfig.h.

12.80.2.3 DciEn

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciEn
```

DCI enable.

Determine if to enable DCI debug from host. **0:Disabled**; 1:Enabled

Definition at line 81 of file DciConfig.h.

12.80.2.4 DciUsb3TypecUfpDbg

```
UINT8 PCH_DCI_PREMEM_CONFIG::DciUsb3TypecUfpDbg
```

USB3 Type-C UFP2DFP kenel / platform debug support.

No change will do nothing to UFP2DFP configuration. When enabled, USB3 Type C UFP (upstream-facing port) may switch to DFP (downstream-facing port) for first connection. It must be enabled for USB3 kernel(kernel mode debug) and platform debug(DFx, DMA, Trace) over UFP Type-C receptacle. Refer to definition of DCI_USB_TYP↩ E_C_DEBUG_MODE for supported settings. 0:Disabled; 1:Enabled; **2:No Change**

Definition at line 96 of file DciConfig.h.

12.80.2.5 KeepEarlyTrace

```
UINT8 PCH_DCI_PREMEM_CONFIG::KeepEarlyTrace
```

Early trace is activated by default.

Enable to keep early trace data and keep tracing; disable to stop tracing; When debug probe is connected and granted, force to no change, let host tool to set it. **0: Disable**; 1: Enable early trace; 2:No Change;

Definition at line 108 of file DciConfig.h.

The documentation for this struct was generated from the following file:

- DciConfig.h

12.81 PCH_DEVICE_INTERRUPT_CONFIG Struct Reference

The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device.

```
#include <InterruptConfig.h>
```

Public Attributes

- UINT8 Device

  Device number.
- UINT8 Function

  Device function.
- UINT8 IntX

  Interrupt pin: INTA-INTD (see PCH_INT_PIN)
- UINT8 Irq

  IRQ to be set for device.

12.81.1 Detailed Description

The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device.

Definition at line 59 of file InterruptConfig.h.

The documentation for this struct was generated from the following file:

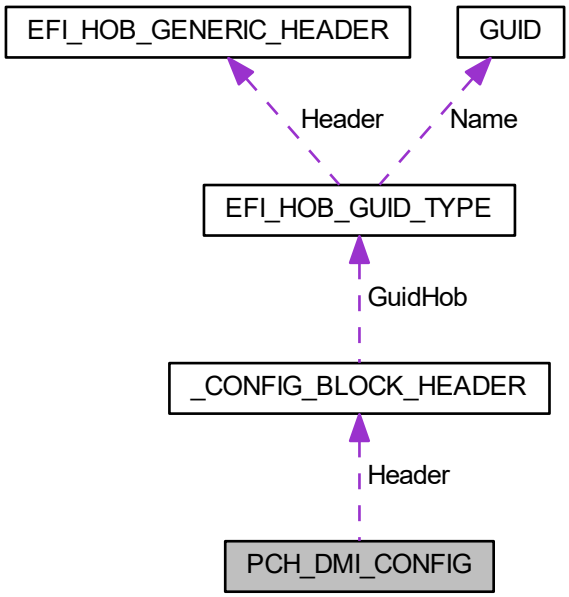
- InterruptConfig.h

12.82 PCH_DMI_CONFIG Struct Reference

The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI.

`#include <PchDmiConfig.h>`

Collaboration diagram for PCH_DMI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PwrOptEnable: 1

  ***0: Disable**; 1: Enable DMI Power Optimizer on PCH side.*
- UINT32 DmiAspmCtrl: 8

  *ASPM configuration on the PCH side of the DMI/OPI Link. Default is **PchPcieAspmAutoConfig***
- UINT32 CwbEnable: 1

  *0: Disable; **1: Enable** Central Write Buffer feature configurable and enabled by default*
- UINT32 L1RpCtl: 1

  *0: Disable; **1: Enable** Allow DMI enter L1 when all root ports are in L1, L0s or link down. Disabled by default.*
- UINT32 DmiPowerReduction: 1

  When set to TRUE turns on:
- UINT32 ClockGating: 1

  0: Disable; 1: Enable clock gating.
- UINT32 Rsvdbits: 19

  Reserved bits.

12.82.1 Detailed Description

The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI.

Revision 1:

- Initial version.

Definition at line 50 of file PchDmiConfig.h.

12.82.2 Member Data Documentation

12.82.2.1 DmiPowerReduction

```
UINT32 PCH_DMI_CONFIG::DmiPowerReduction
```

When set to TRUE turns on:

- L1 State Controller Power Gating
- L1 State PHY Data Lane Power Gating
- PHY Common Lane Power Gating
- Hardware Autonomous Enable
- PMC Request Enable and Sleep Enable

Definition at line 65 of file PchDmiConfig.h.

The documentation for this struct was generated from the following file:

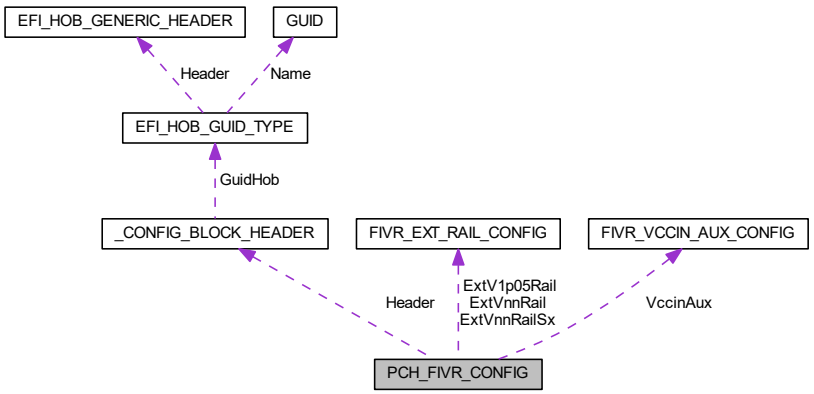
- PchDmiConfig.h

12.83 PCH_FIVR_CONFIG Struct Reference

The PCH_FIVR_CONFIG block describes FIVR settings.

`#include <FivrConfig.h>`

Collaboration diagram for PCH_FIVR_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- FIVR_EXT_RAIL_CONFIG ExtV1p05Rail

  External V1P05 VR rail configuration.
- FIVR_EXT_RAIL_CONFIG ExtVnnRail

  External Vnn VR rail configuration.
- FIVR_EXT_RAIL_CONFIG ExtVnnRailSx

  Additional External Vnn VR rail configuration that will get applied in Sx entry SMI callback.
- FIVR_VCCIN_AUX_CONFIG VccinAux

  VCCIN_AUX voltage rail configuration.
- UINT32 FivrDynPm: 1

  Enable/Disable FIVR Dynamic Power Management Default is ***1*** .

12.83.1 Detailed Description

The PCH_FIVR_CONFIG block describes FIVR settings.

Definition at line 164 of file FivrConfig.h.

12.83.2 Member Data Documentation

12.83.2.1 ExtVnnRailSx

```
FIVR_EXT_RAIL_CONFIG PCH_FIVR_CONFIG::ExtVnnRailSx
```

Additional External Vnn VR rail configuration that will get applied in Sx entry SMI callback.

Required only if External Vnn VR needs different settings for Sx than those specified in ExtVnnRail.

Definition at line 179 of file FivrConfig.h.

The documentation for this struct was generated from the following file:

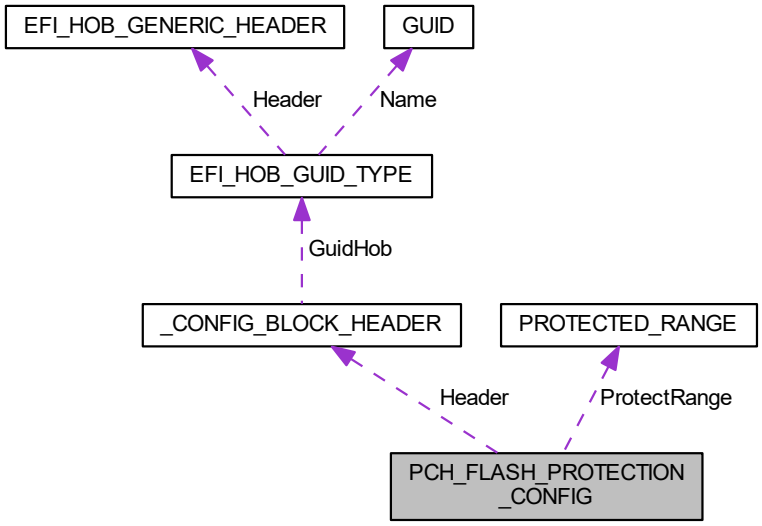
- FivrConfig.h

12.84 PCH_FLASH_PROTECTION_CONFIG Struct Reference

The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled.

```
#include <FlashProtectionConfig.h>
```

Collaboration diagram for PCH_FLASH_PROTECTION_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PROTECTED_RANGE ProtectRange [PCH_FLASH_PROTECTED_RANGES]

  Protected Flash Ranges.

12.84.1 Detailed Description

The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled.

PROTECTED_RANGE is used to specify if flash protection are enabled, the write protection enable bit and the read protection enable bit, and to specify the upper limit and lower base for each register Platform code is responsible to get the range base by PchGetSpiRegionAddresses routine, and set the limit and base accordingly.

Definition at line 76 of file FlashProtectionConfig.h.

The documentation for this struct was generated from the following file:

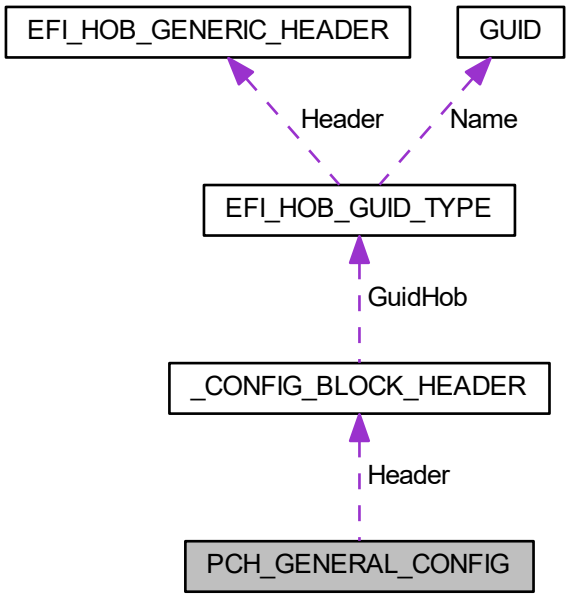
- FlashProtectionConfig.h

12.85 PCH_GENERAL_CONFIG Struct Reference

PCH General Configuration **Revision 2**: - Add VtdEnabled field.

```
#include <PchGeneralConfig.h>
```

Collaboration diagram for PCH_GENERAL_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Crid: 1

  This member describes whether or not the Compatibility Revision ID (CRID) feature of PCH should be enabled.
- UINT32 LegacyIoLowLatency: 1

  Set to enable low latency of legacy IO.
- UINT32 VtdEnabled: 1

  This member describes whether or not the VTD feature should be enabled in PCH.
- UINT32 RsvdBits0: 29

  Reserved bits.

12.85.1 Detailed Description

PCH General Configuration **Revision 2**: - Add VtdEnabled field.

Definition at line 67 of file PchGeneralConfig.h.

12.85.2 Member Data Documentation

12.85.2.1 Crid

```
UINT32 PCH_GENERAL_CONFIG::Crid
```

This member describes whether or not the Compatibility Revision ID (CRID) feature of PCH should be enabled.

0: Disable; 1: Enable

Definition at line 73 of file PchGeneralConfig.h.

12.85.2.2 LegacyIoLowLatency

```
UINT32 PCH_GENERAL_CONFIG::LegacyIoLowLatency
```

Set to enable low latency of legacy IO.

Some systems require lower IO latency irrespective of power. This is a tradeoff between power and IO latency.

Note

: Once this is enabled, DmiAspm, Pcie DmiAspm in SystemAgent and ITSS Clock Gating are forced to disabled. **0: Disable**, 1: Enable

Definition at line 82 of file PchGeneralConfig.h.

12.85.2.3 VtdEnabled

```
UINT32 PCH_GENERAL_CONFIG::VtdEnabled
```

This member describes whether or not the VTD feature should be enabled in PCH.

0: Disable; 1: Enable

Definition at line 87 of file PchGeneralConfig.h.

The documentation for this struct was generated from the following file:

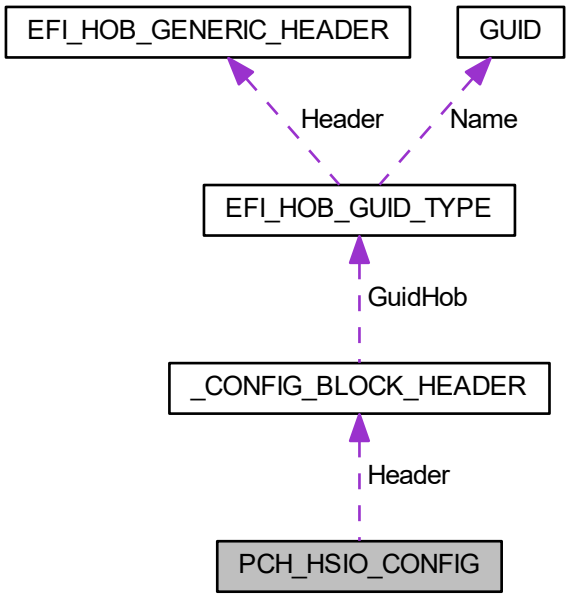
- PchGeneralConfig.h

12.86 PCH_HSIO_CONFIG Struct Reference

The PCH_HSIO_CONFIG block provides HSIO message related settings.

```
#include <HsioConfig.h>
```

Collaboration diagram for PCH_HSIO_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 ChipsetInitBinPtr

  Policy used to point to the Base (+ OEM) ChipsetInit binary used to sync between BIOS and CSME.
- UINT32 ChipsetInitBinLen

  Policy used to indicate the size of the Base (+ OEM) ChipsetInit binary used to sync between BIOS and CSME.
- UINT32 NphyBinPtr

  Policy used to point to the OEM Nphy binary used to sync between BIOS and CSME.
- UINT32 NphyBinLen

  Policy used to indicate the size of the OEM Nphy binary used to sync between BIOS and CSME.
- UINT32 SynpsPhyBinPtr

  Policy used to point to the OEM Synophsys Phy binary used to sync between BIOS and CSME.
- UINT32 SynpsPhyBinLen

  Policy used to indicate the size of the OEM Synophsys Phy binary used to sync between BIOS and CSME.

12.86.1 Detailed Description

The PCH_HSIO_CONFIG block provides HSIO message related settings.

Definition at line 83 of file HsioConfig.h.

The documentation for this struct was generated from the following file:

- HsioConfig.h

12.87 PCH_HSIO_PCIE_LANE_CONFIG Struct Reference

The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane.

```
#include <HsioPcieConfig.h>
```

Public Attributes

- UINT32 HsioRxSetCtleEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 3 Set CTLE Value*
- UINT32 HsioRxSetCtle: 6

  PCH PCIe Gen 3 Set CTLE Value.
- UINT32 HsioTxGen1DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 1 TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen1DownscaleAmp: 6

  PCH PCIe Gen 1 TX Output Downscale Amplitude Adjustment value.
- UINT32 HsioTxGen2DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 2 TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen2DownscaleAmp: 6

  PCH PCIe Gen 2 TX Output Downscale Amplitude Adjustment value.
- UINT32 HsioTxGen3DownscaleAmpEnable: 1

***0: Disable**; 1: Enable PCH PCIe Gen 3 TX Output Downscale Amplitude Adjustment value override*

- UINT32 HsioTxGen3DownscaleAmp: 6

  PCH PCIe Gen 3 TX Output Downscale Amplitude Adjustment value.

- UINT32 RsvdBits0: 4

  Reserved Bits.

- UINT32 HsioTxGen1DeEmphEnable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 1 TX Output De-Emphasis Adjustment Setting value override*

- UINT32 HsioTxGen1DeEmph: 6

  PCH PCIe Gen 1 TX Output De-Emphasis Adjustment Setting.

- UINT32 HsioTxGen2DeEmph3p5Enable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 2 TX Output -3.5dB Mode De-Emphasis Adjustment Setting value override*

- UINT32 HsioTxGen2DeEmph3p5: 6

  PCH PCIe Gen 2 TX Output -3.5dB Mode De-Emphasis Adjustment Setting.

- UINT32 HsioTxGen2DeEmph6p0Enable: 1

  ***0: Disable**; 1: Enable PCH PCIe Gen 2 TX Output -6.0dB Mode De-Emphasis Adjustment Setting value override*

- UINT32 HsioTxGen2DeEmph6p0: 6

  PCH PCIe Gen 2 TX Output -6.0dB Mode De-Emphasis Adjustment Setting.

- UINT32 RsvdBits1: 11

  Reserved Bits.

12.87.1 Detailed Description

The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane.

Definition at line 48 of file HsioPcieConfig.h.

The documentation for this struct was generated from the following file:

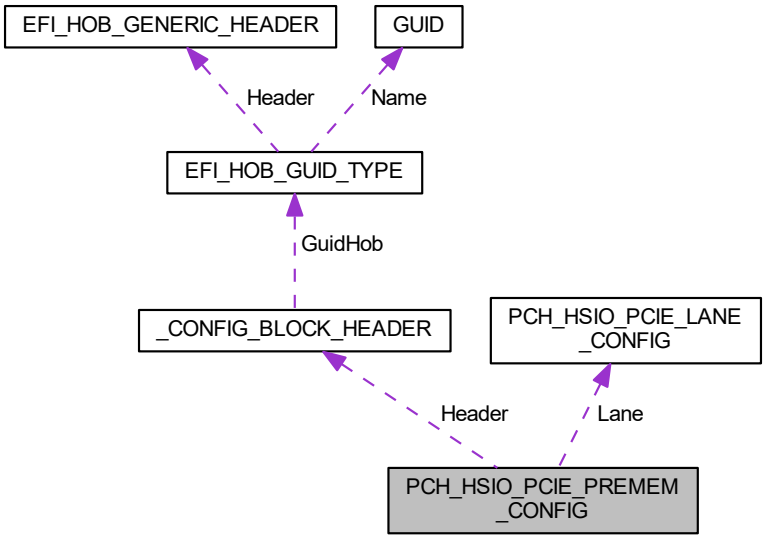
- HsioPcieConfig.h

12.88 PCH_HSIO_PCIE_PREMEM_CONFIG Struct Reference

The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes.

```
#include <HsioPcieConfig.h>
```

Collaboration diagram for PCH_HSIO_PCIE_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCH_HSIO_PCIE_LANE_CONFIG Lane [PCH_MAX_PCIE_ROOT_PORTS]

  These members describe the configuration of HSIO for PCIe lanes.

12.88.1 Detailed Description

The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes.

Definition at line 76 of file HsioPcieConfig.h.

The documentation for this struct was generated from the following file:

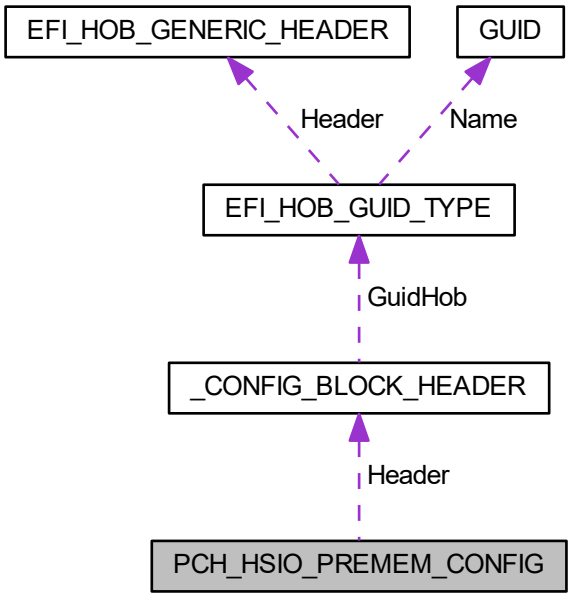
- HsioPcieConfig.h

12.89 PCH_HSIO_PREMEM_CONFIG Struct Reference

The PCH_HSIO_PREMEM_CONFIG block provides HSIO message related settings.

```
#include <HsioConfig.h>
```

Collaboration diagram for PCH_HSIO_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 ChipsetInitMessage

  ***(Test)** 0- Disable, disable will prevent the HSIO version check and ChipsetInit HECI message from being sent **1- Enable** ChipsetInit HECI message*
- UINT8 BypassPhySyncReset

  ***(Test) 0- Disable** 1- Enable When enabled, this is used to bypass the reset after ChipsetInit HECI message.*

12.89.1 Detailed Description

The PCH_HSIO_PREMEM_CONFIG block provides HSIO message related settings.

Definition at line 60 of file HsioConfig.h.

The documentation for this struct was generated from the following file:

- HsioConfig.h

12.90 PCH_HSIO_SATA_PORT_LANE Struct Reference

The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane.

```
#include <HsioSataConfig.h>
```

Public Attributes

- UINT32 HsioRxGen1EqBoostMagEnable: 1

  ***0: Disable**; 1: Enable Receiver Equalization Boost Magnitude Adjustment Value override*
- UINT32 HsioRxGen1EqBoostMag: 6

  SATA 1.5 Gb/sReceiver Equalization Boost Magnitude Adjustment value.
- UINT32 HsioRxGen2EqBoostMagEnable: 1

  ***0: Disable**; 1: Enable Receiver Equalization Boost Magnitude Adjustment Value override*
- UINT32 HsioRxGen2EqBoostMag: 6

  SATA 3.0 Gb/sReceiver Equalization Boost Magnitude Adjustment value.
- UINT32 HsioRxGen3EqBoostMagEnable: 1

  ***0: Disable**; 1: Enable Receiver Equalization Boost Magnitude Adjustment Value override*
- UINT32 HsioRxGen3EqBoostMag: 6

  SATA 6.0 Gb/sReceiver Equalization Boost Magnitude Adjustment value.
- UINT32 HsioTxGen1DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable SATA 1.5 Gb/s TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen1DownscaleAmp: 6

  SATA 1.5 Gb/s TX Output Downscale Amplitude Adjustment value.
- UINT32 RsvdBits0: 4

  Reserved bits.
- UINT32 HsioTxGen2DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable SATA 3.0 Gb/s TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen2DownscaleAmp: 6

  SATA 3.0 Gb/s TX Output Downscale Amplitude Adjustment.
- UINT32 HsioTxGen3DownscaleAmpEnable: 1

  ***0: Disable**; 1: Enable SATA 6.0 Gb/s TX Output Downscale Amplitude Adjustment value override*
- UINT32 HsioTxGen3DownscaleAmp: 6

  SATA 6.0 Gb/s TX Output Downscale Amplitude Adjustment.
- UINT32 HsioTxGen1DeEmphEnable: 1

  ***0: Disable**; 1: Enable SATA 1.5 Gb/s TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen1DeEmph: 6

  SATA 1.5 Gb/s TX Output De-Emphasis Adjustment Setting.
- UINT32 HsioTxGen2DeEmphEnable: 1

  ***0: Disable**; 1: Enable SATA 3.0 Gb/s TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen2DeEmph: 6

  SATA 3.0 Gb/s TX Output De-Emphasis Adjustment Setting.
- UINT32 RsvdBits1: 4

  Reserved bits.
- UINT32 HsioTxGen3DeEmphEnable: 1

  ***0: Disable**; 1: Enable SATA 6.0 Gb/s TX Output De-Emphasis Adjustment Setting value override*
- UINT32 HsioTxGen3DeEmph: 6

  SATA 6.0 Gb/s TX Output De-Emphasis Adjustment Setting value override.
- UINT32 RsvdBits2: 25

  Reserved bits.

12.90.1 Detailed Description

The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane.

Definition at line 46 of file HsioSataConfig.h.

The documentation for this struct was generated from the following file:

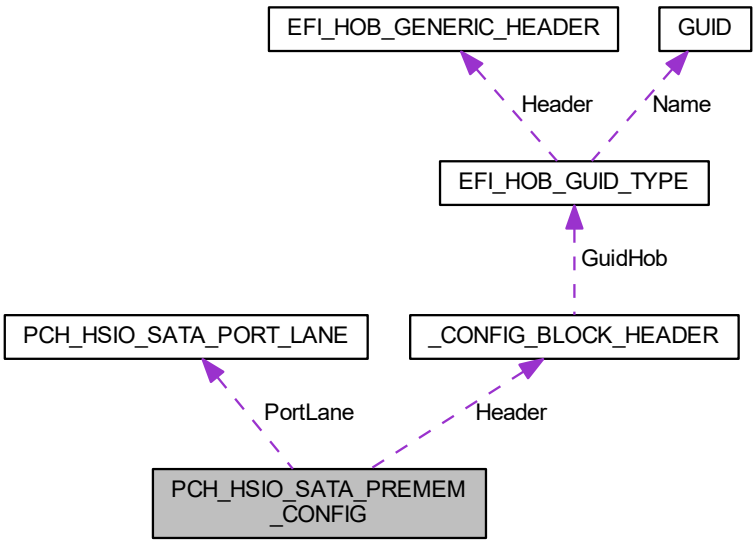
- HsioSataConfig.h

12.91 PCH_HSIO_SATA_PREMEM_CONFIG Struct Reference

The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller.

`#include <HsioSataConfig.h>`

Collaboration diagram for PCH_HSIO_SATA_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCH_HSIO_SATA_PORT_LANE PortLane [PCH_MAX_SATA_PORTS]

  These members describe the configuration of HSIO for SATA lanes.

12.91.1 Detailed Description

The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller.

Definition at line 82 of file HsioSataConfig.h.

The documentation for this struct was generated from the following file:

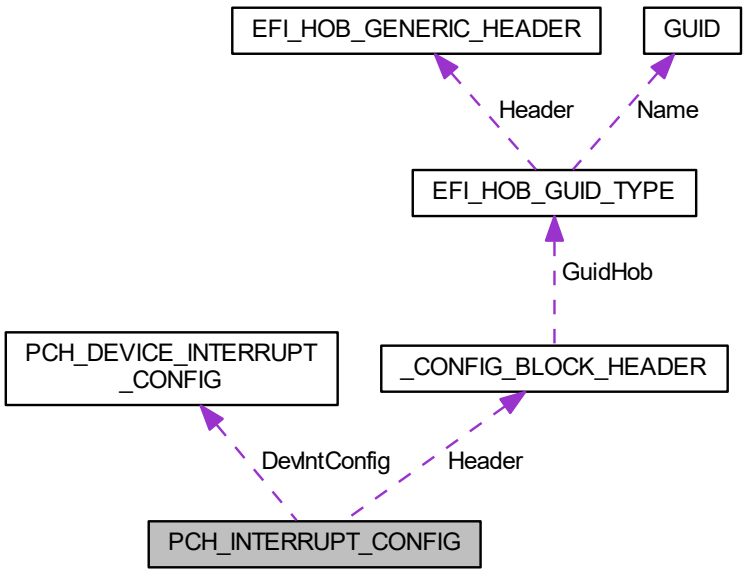
- HsioSataConfig.h

12.92 PCH_INTERRUPT_CONFIG Struct Reference

The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH.

```
#include <InterruptConfig.h>
```

Collaboration diagram for PCH_INTERRUPT_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 NumOfDevIntConfig

  Number of entries in DevIntConfig table.
- UINT8 Rsvd0 [3]

  Reserved bytes, align to multiple 4.
- PCH_DEVICE_INTERRUPT_CONFIG DevIntConfig [PCH_MAX_DEVICE_INTERRUPT_CONFIG]

  Array which stores PCH devices interrupts settings.
- UINT8 GpioIrqRoute

  *Interrupt routing for GPIO. Default is **14**.*
- UINT8 SciIrqSelect

  *Interrupt select for SCI. Default is **9**.*
- UINT8 TcoIrqSelect

  *Interrupt select for TCO. Default is **9**.*
- UINT8 TcoIrqEnable

  *Enable IRQ generation for TCO. **0: Disable**; 1: Enable.*

12.92.1 Detailed Description

The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH.

Definition at line 75 of file InterruptConfig.h.

The documentation for this struct was generated from the following file:

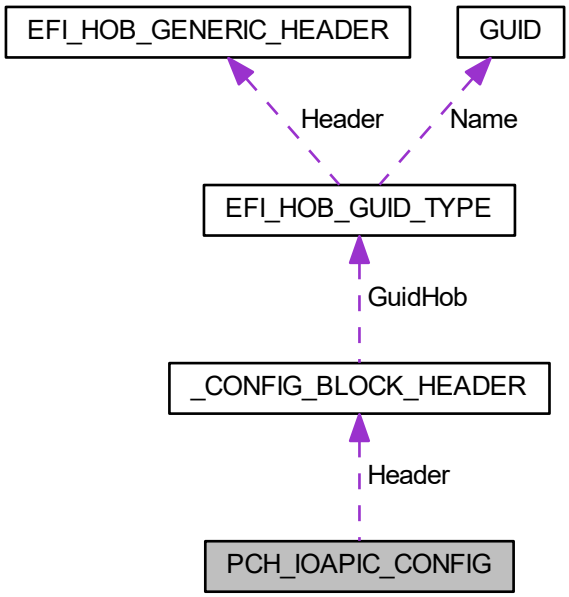
- InterruptConfig.h

12.93 PCH_IOAPIC_CONFIG Struct Reference

The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE.

```
#include <IoApicConfig.h>
```

Collaboration diagram for PCH_IOAPIC_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 IoApicEntry24_119: 1

  *0: Disable; **1: Enable** IOAPIC Entry 24-119*
- UINT32 Enable8254ClockGating: 1

  Enable 8254 Static Clock Gating during early POST time.
- UINT32 Enable8254ClockGatingOnS3: 1

  Enable 8254 Static Clock Gating on S3 resume path.
- UINT32 RsvdBits1: 29

  Reserved bits.
- UINT8 IoApicId

  *This member determines IOAPIC ID. Default is **0x02**.*
- UINT8 Rsvd0 [3]

  Reserved bytes.

12.93.1 Detailed Description

The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE.

Bus:device:function fields will be programmed to the register P2SB IBDF(P2SB PCI offset R6Ch-6Dh), it's using for the following purpose: As the Requester ID when initiating Interrupt Messages to the processor. As the Completer ID when responding to the reads targeting the IOxAPI's Memory-Mapped I/O registers. This field defaults to Bus 0: Device 31: Function 0 after reset. BIOS can program this field to provide a unique Bus:Device:Function number for the internal IOxAPIC. The address resource range of IOAPIC must be reserved in E820 and ACPI as system resource.

Definition at line 57 of file IoApicConfig.h.

12.93.2 Member Data Documentation

12.93.2.1 Enable8254ClockGating

```
UINT32 PCH_IOAPIC_CONFIG::Enable8254ClockGating
```

Enable 8254 Static Clock Gating during early POST time.

0: Disable, **1: Enable** Setting 8254CGE is required to support SLP_S0. Enable this if 8254 timer is not used. However, set 8254CGE=1 in POST time might fail to boot legacy OS using 8254 timer. Make sure it is disabled to support legacy OS using 8254 timer.

Note

: For some OS environment that it needs to set 8254CGE in late state it should set this policy to FALSE and use ItssSet8254ClockGateState (TRUE) in SMM later. This is also required during S3 resume. To avoid SMI requirement in S3 reusme path, it can enable the Enable8254ClockGatingOnS3 and RC will do 8254 CGE programming in PEI during S3 resume with BOOT_SAI.

Definition at line 73 of file IoApicConfig.h.

12.93.2.2 Enable8254ClockGatingOnS3

```
UINT32 PCH_IOAPIC_CONFIG::Enable8254ClockGatingOnS3
```

Enable 8254 Static Clock Gating on S3 resume path.

0: Disable, **1: Enable** This is only applicable when Enable8254ClockGating is disabled. If Enable8254ClockGating is enabled, RC will do the 8254 CGE programming on S3 resume path as well.

Definition at line 80 of file IoApicConfig.h.

The documentation for this struct was generated from the following file:

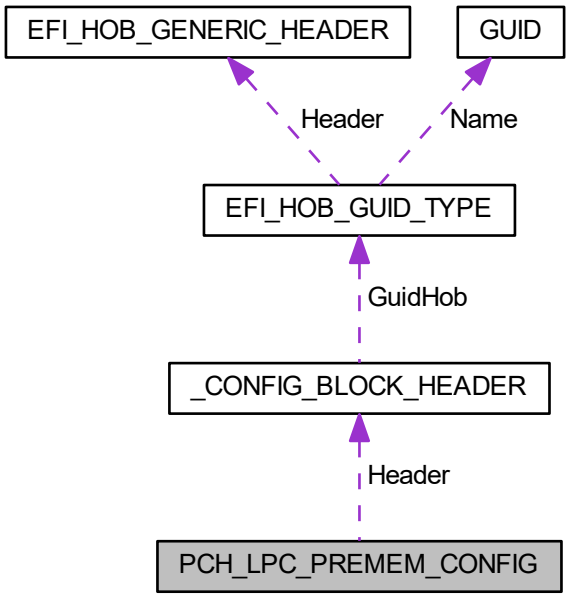
- IoApicConfig.h

12.94 PCH_LPC_PREMEM_CONFIG Struct Reference

This structure contains the policies which are related to LPC.

```
#include <LpcConfig.h>
```

Collaboration diagram for PCH_LPC_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 EnhancePort8xhDecoding: 1

  Enhance the port 8xh decoding.
- UINT32 LpcPmHAE: 1

  Hardware Autonomous Enable.
- UINT32 RsvdBits: 30

  Reserved bits.

12.94.1 Detailed Description

This structure contains the policies which are related to LPC.

Definition at line 46 of file LpcConfig.h.

12.94.2 Member Data Documentation

12.94.2.1 EnhancePort8xhDecoding

```
UINT32 PCH_LPC_PREMEM_CONFIG::EnhancePort8xhDecoding
```

Enhance the port 8xh decoding.

Original LPC only decodes one byte of port 80h, with this enhancement LPC can decode word or dword of port 80h-83h.

Note

: this will occupy one LPC generic IO range register. While this is enabled, read from port 80h always return 0x00. 0: Disable, **1: Enable**

Definition at line 54 of file LpcConfig.h.

12.94.2.2 LpcPmHAE

```
UINT32 PCH_LPC_PREMEM_CONFIG::LpcPmHAE
```

Hardware Autonomous Enable.

When enabled, LPC will automatically engage power gating when it has reached its idle condition. 0: Disable, **1: Enable**

Definition at line 60 of file LpcConfig.h.

The documentation for this struct was generated from the following file:

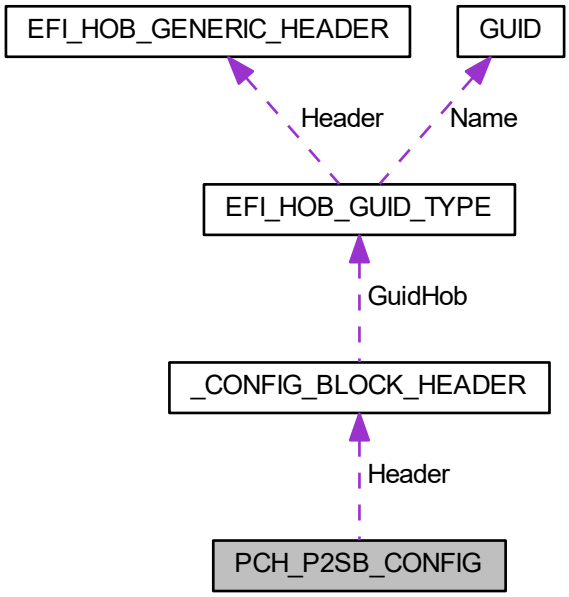
- LpcConfig.h

12.95 PCH_P2SB_CONFIG Struct Reference

This structure contains the policies which are related to P2SB device.

`#include <P2sbConfig.h>`

Collaboration diagram for PCH_P2SB_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 SbAccessUnlock: 1

  (Test) *The sideband MMIO register access to specific ports will be locked before 3rd party code execution.*
- UINT32 Rsvdbits: 31

  Reserved bits.

12.95.1 Detailed Description

This structure contains the policies which are related to P2SB device.

Definition at line 46 of file P2sbConfig.h.

12.95.2 Member Data Documentation

12.95.2.1 SbAccessUnlock

```
UINT32 PCH_P2SB_CONFIG::SbAccessUnlock
```

(Test) The sideband MMIO register access to specific ports will be locked before 3rd party code execution.

Currently it disables PSFx access. This policy unlocks the sideband MMIO space for those IPs. **0: Lock sideband access** ; 1: Unlock sideband access. NOTE: Do not set this policy "SbAccessUnlock" unless its necessary.

Definition at line 56 of file P2sbConfig.h.

The documentation for this struct was generated from the following file:

- P2sbConfig.h

12.96 PCH_PCIE_CLOCK Struct Reference

PCH_PCIE_CLOCK describes PCIe source clock generated by PCH.

```
#include <PchPcieRpConfig.h>
```

Public Attributes

- UINT8 Usage

  Purpose of given clock (see PCH_PCIE_CLOCK_USAGE). Default: Unused, 0xFF.
- UINT8 ClkReq

  ClkSrc - ClkReq mapping. Default: 1:1 mapping with Clock numbers.
- UINT8 RsvdBytes [2]

  Reserved byte.

12.96.1 Detailed Description

PCH_PCIE_CLOCK describes PCIe source clock generated by PCH.

Definition at line 159 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

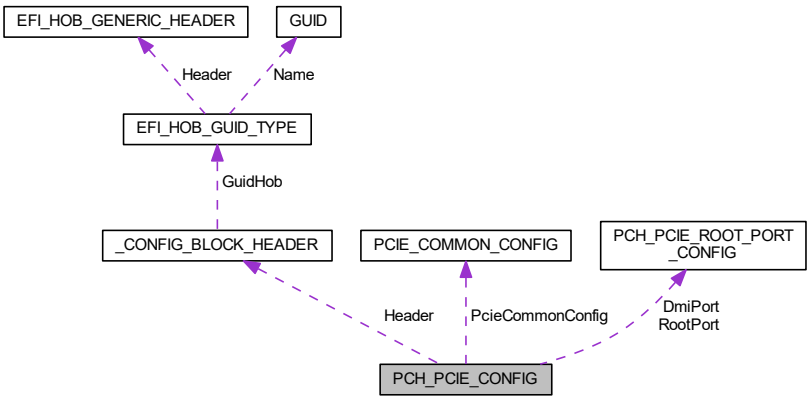
- PchPcieRpConfig.h

12.97 PCH_PCIE_CONFIG Struct Reference

The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**:

```
#include <PchPcieRpConfig.h>
```

Collaboration diagram for PCH_PCIE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCIE_COMMON_CONFIG PcieCommonConfig

  These members describe the configuration of each PCH PCIe root port.
- UINT8 PchPciePort8xhDecodePortIndex

  ***(Test)** The Index of PCIe Port that is selected for Port8xh Decode (0 Based)*

12.97.1 Detailed Description

The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**:

- Initial version. **Revision 2**:
- Moved EnablePort8xhDecode policy to PCIE_COMMON_CONFIG **Revision 3**:
- Add DmiPowerGatingDis

Definition at line 240 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

- PchPcieRpConfig.h

12.98 PCH_PCIE_ROOT_PORT_CONFIG Struct Reference

The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port.

`#include <PchPcieRpConfig.h>`

Public Attributes

- UINT8 ExtSync

  an instance of Pcie Common Config
- UINT8 SystemErrorEnable

  *Indicate whether the System Error is enabled. **0: Disable**; 1: Enable.*
- UINT8 MvcEnabled

  The Multiple VC (MVC) supports hardware to avoid HoQ block for latency sensitive TC.
- UINT8 VppPort

  Virtual Pin Port is industry standard introduced to PCIe Hot Plug support in systems when GPIO pins expansion is needed.
- UINT8 VppAddress

  PCIe Hot Plug VPP SMBus Address. Default is zero.
- UINT8 RsvdBytes0 [3]

  Reserved bytes.

12.98.1 Detailed Description

The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port.

Definition at line 201 of file PchPcieRpConfig.h.

12.98.2 Member Data Documentation

12.98.2.1 ExtSync

`UINT8 PCH_PCIE_ROOT_PORT_CONFIG::ExtSync`

an instance of Pcie Common Config

Indicate whether the extended synch is enabled. **0: Disable**; 1: Enable.

Definition at line 203 of file PchPcieRpConfig.h.

12.98.2.2 MvcEnabled

`UINT8 PCH_PCIE_ROOT_PORT_CONFIG::MvcEnabled`

The Multiple VC (MVC) supports hardware to avoid HoQ block for latency sensitive TC.

Currently it is only applicable to Root Ports with 2pX4 port configuration with 2 VCs,or DMI port configuration with 3 VCs. For Root Ports 2pX4 configuration, two RPs (RP0, RP2) shall support two PCIe VCs (VC0 & VC1) and the other RPs (RP1, RP3) shall be disabled. **0: Disable**; 1: Enable

Definition at line 216 of file PchPcieRpConfig.h.

12.98.2.3 VppPort

`UINT8 PCH_PCIE_ROOT_PORT_CONFIG::VppPort`

Virtual Pin Port is industry standard introduced to PCIe Hot Plug support in systems when GPIO pins expansion is needed.

It is server specific feature. **0x00: Default**; 0xFF: Disabled

Definition at line 222 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

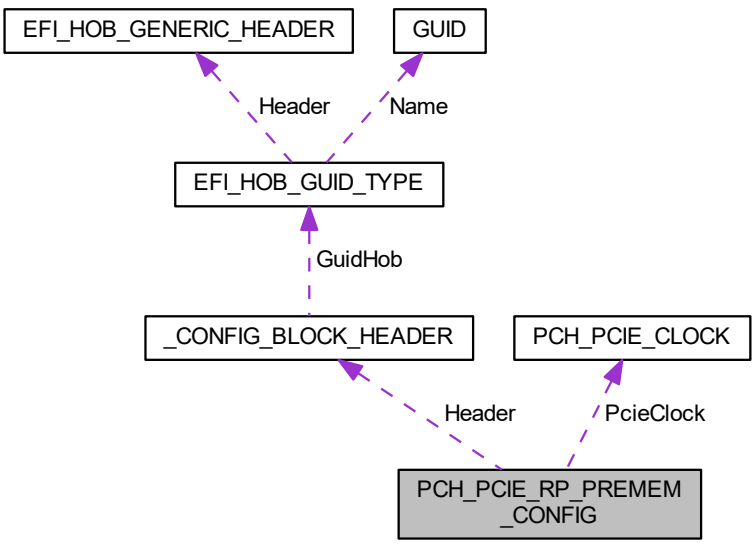
- PchPcieRpConfig.h

12.99 PCH_PCIE_RP_PREMEM_CONFIG Struct Reference

The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**:

`#include <PchPcieRpConfig.h>`

Collaboration diagram for PCH_PCIE_RP_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 RpEnabledMask

  Root Port enabling mask.
- PCH_PCIE_CLOCK PcieClock [PCH_MAX_PCIE_CLOCKS]

  Configuration of PCIe source clocks.
- UINT8 Bifurcation [PCH_MAX_PCIE_CONTROLLERS]

  *Per Controller Bifurcation Configuration **0: Disabled**; 1: 4x1; 2: 1x2_2x1; 3: 2x2; 4: 1x4; 5: 4x2; 6: 1x4_2x2; 7: 2x2_1x4; 8: 2x4; 9: 1x8 (see: PCIE_BIFURCATION_CONFIG)*

12.99.1 Detailed Description

The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**:

- Initial version.

Definition at line 263 of file PchPcieRpConfig.h.

12.99.2 Member Data Documentation

12.99.2.1 RpEnabledMask

```
UINT32 PCH_PCIE_RP_PREMEM_CONFIG::RpEnabledMask
```

Root Port enabling mask.

Bit0 presents RP1, Bit1 presents RP2, and so on. 0: Disable; **1: Enable**.

Definition at line 270 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

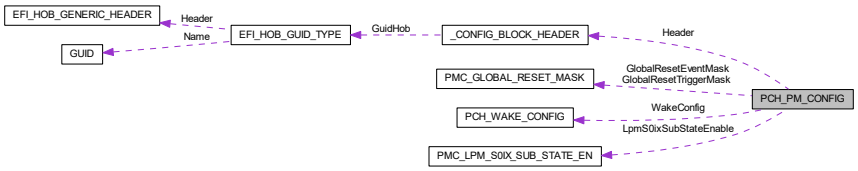
- PchPcieRpConfig.h

12.100 PCH_PM_CONFIG Struct Reference

The PCH_PM_CONFIG block describes expected miscellaneous power management settings.

```
#include <PmConfig.h>
```

Collaboration diagram for PCH_PM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCH_WAKE_CONFIG WakeConfig

  Specify Wake Policy.
- UINT32 PchDeepSxPol: 4

  Deep Sx Policy. Refer to PCH_DEEP_SX_CONFIG for each value. Default is ***PchDeepSxPolDisable****.*
- UINT32 PchSlpS3MinAssert: 4

  SLP_S3 Minimum Assertion Width Policy. Refer to PCH_SLP_S3_MIN_ASSERT for each value. Default is ***Pch↩SlpS350ms****.*
- UINT32 PchSlpS4MinAssert: 4

  SLP_S4 Minimum Assertion Width Policy. Refer to PCH_SLP_S4_MIN_ASSERT for each value. Default is ***Pch↩SlpS44s****.*
- UINT32 PchSlpSusMinAssert: 4

  SLP_SUS Minimum Assertion Width Policy. Refer to PCH_SLP_SUS_MIN_ASSERT for each value. Default is ***Pch↩SlpSus4s****.*
- UINT32 PchSlpAMinAssert: 4

  SLP_A Minimum Assertion Width Policy. Refer to PCH_SLP_A_MIN_ASSERT for each value. Default is ***PchSlpA2s****.*
- UINT32 SlpStrchSusUp: 1

  This member describes whether or not the LPC ClockRun feature of PCH should be enabled.
- UINT32 SlpLanLowDc: 1

  Enable/Disable SLP_LAN# Low on DC Power.
- UINT32 PwrBtnOverridePeriod: 3

  PCH power button override period.
- UINT32 DisableEnergyReport: 1

  ***(Test)** Disable/Enable PCH to CPU enery report feature.*
- UINT32 DisableDsxAcPresentPulldown: 1

  When set to Disable, PCH will internal pull down AC_PRESENT in deep SX and during G3 exit.
- UINT32 DisableNativePowerButton: 1

  Power button native mode disable.
- UINT32 MeWakeSts: 1

  Clear the ME_WAKE_STS bit in the Power and Reset Status (PRSTS) register. 0: Disable; ***1: Enable****.*
- UINT32 WolOvrWkSts: 1

  Clear the WOL_OVR_WK_STS bit in the Power and Reset Status (PRSTS) register. 0: Disable; ***1: Enable****.*
- UINT32 PsOnEnable: 1

  Decide if PS_ON is to be enabled.
- UINT32 CppmFaEn: 1

  Decide if CPPM Force Alignment is to be enabled.
- UINT32 CpuC10GatePinEnable: 1

  Enable/Disable platform support for CPU_C10_GATE# pin to control gating of CPU VccIO and VccSTG rails instead of SLP_S0# pin.
- UINT32 PmcDbgMsgEn: 1

  Control whether to enable PMC debug messages to Trace Hub.
- UINT32 ModPhySusPgEnable: 1

  Enable/Disable ModPHY SUS Power Domain Dynamic Gating.
- UINT32 Usb2PhySusPgEnable: 1

  ***(Test)** This policy option enables USB2 PHY SUS Well Power Gating functionality.*
- UINT32 OsIdleEnable: 1

  Enable Os Idle Mode.
- UINT32 V1p05PhyExtFetControlEn: 1

  Enable control using EXT_PWR_GATE# pin of external FET to power gate v1p05-PHY ***0: Disable****; 1: Enable.*

- UINT32 V1p05IsExtFetControlEn: 1

  *Enable control using EXT_PWR_GATE2# pin of external FET to power gate v1p05-IS supply **0: Disable**; 1: Enable.*
- UINT32 S0ixAutoDemotion: 1

  Enable/Disable the Low Power Mode Host S0ix Auto-Demotion feature.
- UINT32 LatchEventsC10Exit: 1

  Enable/Disable Latch Events C10 Exit.
- UINT32 PmcWdtTimerEn: 1

  Enable/Disable PMC Watch Dog Timer.
- UINT8 PchPwrCycDur

  Reset Power Cycle Duration could be customized in the unit of second.
- UINT8 PciePllSsc

  Specifies the Pcie Pll Spread Spectrum Percentage The value of this policy is in 1/10th percent units.
- UINT8 C10DynamicThresholdAdjustment

  Tells BIOS to enable C10 dynamic threshold adjustment mode.
- UINT8 Rsvd0 [1]

  Reserved bytes.
- PMC_LPM_S0IX_SUB_STATE_EN LpmS0ixSubStateEnable

  ***(Test)** Low Power Mode Enable/Disable config.*
- UINT8 GlobalResetMasksOverride

  Set true to enable override of Global Reset Event/Trigger masks.
- UINT8 Rsvd1 [3]

  Reserved bytes.
- UINT32 ThermTimerDelay: 10

  PMC Therm Timer Delay Register (THERM_TIMER_DELAY).
- UINT32 PmErDebugMode: 1

  Energy Reporting Debug Mode.
- UINT32 PlatformAtxTelemetryUnit: 1

  *PMC_SUS_SPARE_GCR_0 bit 0, set by BIOS to indicate the telemetry type for Desktop Psys support. Set ATX Telemetry Unit in Watts or Percentage Default value is Watts; **0: Watts**; 1: Percent.*

12.100.1 Detailed Description

The PCH_PM_CONFIG block describes expected miscellaneous power management settings.

The PowerResetStatusClear field would clear the Power/Reset status bits, please set the bits if you want PCH Init driver to clear it, if you want to check the status later then clear the bits.

Revision 1:

- Initial version. **Revision 2**
- Added C10DynamicThresholdAdjustment **Revision 3**
- Added PlatformAtxTelemetryUnit

Definition at line 240 of file PmConfig.h.

12.100.2 Member Data Documentation

12.100.2.1 C10DynamicThresholdAdjustment

```
UINT8 PCH_PM_CONFIG::C10DynamicThresholdAdjustment
```

Tells BIOS to enable C10 dynamic threshold adjustment mode.

BIOS will only attemt to enable it on PCH SKUs which support it.

Definition at line 434 of file PmConfig.h.

12.100.2.2 CppmFaEn

```
UINT32 PCH_PM_CONFIG::CppmFaEn
```

Decide if CPPM Force Alignment is to be enabled.

When enabled, PMC allows stalling of the backbone or blocking the DMI transmit arbiter. 0: Disable; **1: Enable**

Definition at line 325 of file PmConfig.h.

12.100.2.3 CpuC10GatePinEnable

```
UINT32 PCH_PM_CONFIG::CpuC10GatePinEnable
```

Enable/Disable platform support for CPU_C10_GATE# pin to control gating of CPU VccIO and VccSTG rails instead of SLP_S0# pin.

This policy needs to be set if board design includes support for CPU_C10_GATE# pin. 0: Disable; **1: Enable**

Definition at line 332 of file PmConfig.h.

12.100.2.4 DisableDsxAcPresentPulldown

```
UINT32 PCH_PM_CONFIG::DisableDsxAcPresentPulldown
```

When set to Disable, PCH will internal pull down AC_PRESENT in deep SX and during G3 exit.

When set to Enable, PCH will not pull down AC_PRESENT. This setting is ignored when DeepSx is not supported. Default is **0:Disable**

Definition at line 287 of file PmConfig.h.

12.100.2.5 DisableEnergyReport

```
UINT32 PCH_PM_CONFIG::DisableEnergyReport
```

(Test) Disable/Enable PCH to CPU enery report feature.

0: Disable; 1: Enable. Enery Report is must have feature. Wihtout Energy Report, the performance report by workloads/benchmarks will be unrealistic because PCH's energy is not being accounted in power/performance management algorithm. If for some reason PCH energy report is too high, which forces CPU to try to reduce its power by throttling, then it could try to disable Energy Report to do first debug. This might be due to energy scaling factors are not correct or the LPM settings are not kicking in.

Definition at line 280 of file PmConfig.h.

12.100.2.6 DisableNativePowerButton

```
UINT32 PCH_PM_CONFIG::DisableNativePowerButton
```

Power button native mode disable.

While FALSE, the PMC's power button logic will act upon the input value from the GPIO unit, as normal. While TRUE, this will result in the PMC logic constantly seeing the power button as de-asserted. **Default is FALSE.**

Definition at line 294 of file PmConfig.h.

12.100.2.7 GlobalResetMasksOverride

```
UINT8 PCH_PM_CONFIG::GlobalResetMasksOverride
```

Set true to enable override of Global Reset Event/Trigger masks.

Values from GlobalResetTriggerMask and GlobalResetEventMask will be used as override value. **0: Disable**, 1: Enable

Definition at line 473 of file PmConfig.h.

12.100.2.8 LatchEventsC10Exit

```
UINT32 PCH_PM_CONFIG::LatchEventsC10Exit
```

Enable/Disable Latch Events C10 Exit.

When this bit is set to 1, SLP_S0# entry events in SLP_S0_DEBUG_REGx registers are captured on C10 exit (instead of C10 entry which is default) **0: Disable**; 1: Enable.

Definition at line 388 of file PmConfig.h.

12.100.2.9 LpmS0ixSubStateEnable

`PMC_LPM_S0IX_SUB_STATE_EN PCH_PM_CONFIG::LpmS0ixSubStateEnable`

(Test) Low Power Mode Enable/Disable config.

Configure if respective S0i2/3 sub-states are to be supported by the platform. By default all sub-states are enabled but for test purpose respective states can be disabled. **Default is 0xFF**

Definition at line 444 of file PmConfig.h.

12.100.2.10 ModPhySusPgEnable

`UINT32 PCH_PM_CONFIG::ModPhySusPgEnable`

Enable/Disable ModPHY SUS Power Domain Dynamic Gating.

EXT_PWR_GATE# signal (if supported on platform) can be used to control external FET for power gating ModPHY

Note

: This setting is not supported and ignored on PCH-H 0: Disable; **1: Enable**.

Definition at line 348 of file PmConfig.h.

12.100.2.11 OsIdleEnable

`UINT32 PCH_PM_CONFIG::OsIdleEnable`

Enable Os Idle Mode.

0: Disable; **1: Enable**.

Definition at line 361 of file PmConfig.h.

12.100.2.12 PchPwrCycDur

`UINT8 PCH_PM_CONFIG::PchPwrCycDur`

Reset Power Cycle Duration could be customized in the unit of second.

Please refer to EDS for all support settings. PCH HW default is 4 seconds, and range is 1∼4 seconds, where **0 is default**, 1 is 1 second, 2 is 2 seconds, ... 4 is 4 seconds. And make sure the setting correct, which never less than the following register.

- GEN_PMCON_B.SLP_S3_MIN_ASST_WDTH
- GEN_PMCON_B.SLP_S4_MIN_ASST_WDTH
- PWRM_CFG.SLP_A_MIN_ASST_WDTH
- PWRM_CFG.SLP_LAN_MIN_ASST_WDTH

Definition at line 421 of file PmConfig.h.

12.100.2.13 PciePllSsc

`UINT8 PCH_PM_CONFIG::PciePllSsc`

Specifies the Pcie Pll Spread Spectrum Percentage The value of this policy is in 1/10th percent units.

Valid spread range is 0-20. A value of 0xFF is reserved for AUTO. A value of 0 is SSC of 0.0%. A value of 20 is SSC of 2.0% The default is **0xFF: AUTO - No BIOS override**.

Definition at line 429 of file PmConfig.h.

12.100.2.14 PmcDbgMsgEn

`UINT32 PCH_PM_CONFIG::PmcDbgMsgEn`

Control whether to enable PMC debug messages to Trace Hub.

When Enabled, PMC HW will send debug messages to trace hub; When Disabled, PMC HW will never send debug meesages to trace hub.

Note

: When enabled, system may not enter S0ix **0: Disable**; 1: Enable.

Definition at line 340 of file PmConfig.h.

12.100.2.15 PmcWdtTimerEn

`UINT32 PCH_PM_CONFIG::PmcWdtTimerEn`

Enable/Disable PMC Watch Dog Timer.

When this bit is set to 1, timers are enabled that can trigger a system reset. 0: Disable; **1: Enable**.

Definition at line 394 of file PmConfig.h.

12.100.2.16 PsOnEnable

`UINT32 PCH_PM_CONFIG::PsOnEnable`

Decide if PS_ON is to be enabled.

This is available on desktop only. PS_ON is a new C10 state from the CPU on desktop SKUs that enables a lower power target that will be required by the California Energy Commission (CEC). When FALSE, PS_ON is to be disabled.} **0: Disable**; 1: Enable.

Definition at line 319 of file PmConfig.h.

12.100.2.17 PwrBtnOverridePeriod

```
UINT32 PCH_PM_CONFIG::PwrBtnOverridePeriod
```

PCH power button override period.

000b-4s, 001b-6s, 010b-8s, 011b-10s, 100b-12s, 101b-14s **Default is 0: 4s**

Definition at line 268 of file PmConfig.h.

12.100.2.18 S0ixAutoDemotion

```
UINT32 PCH_PM_CONFIG::S0ixAutoDemotion
```

Enable/Disable the Low Power Mode Host S0ix Auto-Demotion feature.

This feature enables the PMC to autonomously manage the deepest allowed S0ix substate to combat thrashing between power management states. 0: Disable; **1: Enable**.

Definition at line 381 of file PmConfig.h.

12.100.2.19 SlpLanLowDc

```
UINT32 PCH_PM_CONFIG::SlpLanLowDc
```

Enable/Disable SLP_LAN# Low on DC Power.

0: Disable; **1: Enable**. Configure On DC PHY Power Diable according to policy SlpLanLowDc. When this is enabled, SLP_LAN# will be driven low when ACPRESENT is low. This indicates that LAN PHY should be powered off on battery mode. This will override the DC_PP_DIS setting by WolEnableOverride.

Definition at line 262 of file PmConfig.h.

12.100.2.20 SlpStrchSusUp

```
UINT32 PCH_PM_CONFIG::SlpStrchSusUp
```

This member describes whether or not the LPC ClockRun feature of PCH should be enabled.

0: Disable; 1: Enable **0: Disable**; 1: Enable SLP_X Stretching After SUS Well Power Up

Definition at line 254 of file PmConfig.h.

12.100.2.21 ThermTimerDelay

```
UINT32 PCH_PM_CONFIG::ThermTimerDelay
```

PMC Therm Timer Delay Register (THERM_TIMER_DELAY).

Unit in us(microseconds). Adding delay time between CPU thermal trip propagating through PCH and PCH generating a Global Reset **Default is 500**

Definition at line 488 of file PmConfig.h.

12.100.2.22 Usb2PhySusPgEnable

```
UINT32 PCH_PM_CONFIG::Usb2PhySusPgEnable
```

(Test) This policy option enables USB2 PHY SUS Well Power Gating functionality.

Note

: This setting is not supported and ignored on PCH-H 0: disable USB2 PHY SUS Well Power Gating **1: enable USB2 PHY SUS Well Power Gating**

Definition at line 356 of file PmConfig.h.

The documentation for this struct was generated from the following file:

- PmConfig.h

12.101 PCH_SATA_PORT_CONFIG Struct Reference

This structure configures the features, property, and capability for each SATA port.

```
#include <SataConfig.h>
```

Public Attributes

- UINT32 Enable: 1

  Enable SATA port.
- UINT32 HotPlug: 1

  ***0: Disable**; 1: Enable*
- UINT32 InterlockSw: 1

  ***0: Disable**; 1: Enable*
- UINT32 External: 1

  ***0: Disable**; 1: Enable*
- UINT32 SpinUp: 1

  ***0: Disable**; 1: Enable the COMRESET initialization Sequence to the device*
- UINT32 SolidStateDrive: 1

  ***0: HDD**; 1: SSD*
- UINT32 DevSlp: 1

  ***0: Disable**; 1: Enable DEVSLP on the port*
- UINT32 EnableDitoConfig: 1

  ***0: Disable**; 1: Enable DEVSLP Idle Timeout settings (DmVal, DitoVal)*
- UINT32 DmVal: 4

  *DITO multiplier. Default is **15**.*
- UINT32 DitoVal: 10

  *DEVSLP Idle Timeout (DITO), Default is **625**.*
- UINT32 ZpOdd: 1

  *Support zero power ODD **0: Disable**, 1: Enable.*
- UINT32 DevSlpResetConfig: 4

  *0: Hardware default; **0x01: GpioResumeReset**; 0x03: GpioHostDeepReset; 0x05: GpioPlatformReset; 0x07↩ : GpioDswReset*
- UINT32 RxPolarity: 1

  ***0: Disable**; 1: Enable; Rx Polarity*
- UINT32 RsvdBits0: 4

  Reserved fields for future expansion w/o protocol change.

12.101.1 Detailed Description

This structure configures the features, property, and capability for each SATA port.

Definition at line 75 of file SataConfig.h.

12.101.2 Member Data Documentation

12.101.2.1 Enable

```
UINT32 PCH_SATA_PORT_CONFIG::Enable
```

Enable SATA port.

It is highly recommended to disable unused ports for power savings 0: Disable; **1: Enable**

Definition at line 80 of file SataConfig.h.

12.101.2.2 ZpOdd

```
UINT32 PCH_SATA_PORT_CONFIG::ZpOdd
```

Support zero power ODD **0: Disable**, 1: Enable.

This is also used to disable ModPHY dynamic power gate.

Definition at line 94 of file SataConfig.h.

The documentation for this struct was generated from the following file:

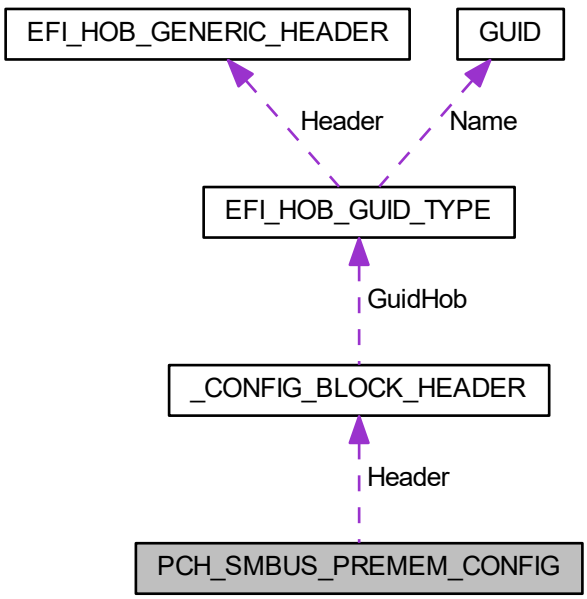
- SataConfig.h

12.102 PCH_SMBUS_PREMEM_CONFIG Struct Reference

The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform.

```
#include <SmbusConfig.h>
```

Collaboration diagram for PCH_SMBUS_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Revision 1: Init version.
- UINT32 Enable: 1

  This member describes whether or not the SMBus controller of PCH should be enabled.
- UINT32 ArpEnable: 1

  *Enable SMBus ARP support, **0: Disable**; 1: Enable.*
- UINT32 DynamicPowerGating: 1

  ***(Test) Disable** or Enable Smbus dynamic power gating.*
- UINT32 SpdWriteDisable: 1

  (Test)** SPD Write Disable, 0: leave SPD Write Disable bit; **1: set SPD Write Disable bit.
- UINT32 SmbAlertEnable: 1

  *Enable SMBus Alert pin (SMBALERT#). 0: **Disabled, 1: Enabled.***
- UINT32 RsvdBits0: 27

  Reserved bits.
- UINT16 SmbusIoBase

  *SMBUS Base Address (IO space). Default is **0xEFA0**.*
- UINT8 Rsvd0

  Reserved bytes.
- UINT8 NumRsvdSmbusAddresses

  The number of elements in the RsvdSmbusAddressTable.
- UINT8 RsvdSmbusAddressTable [PCH_MAX_SMBUS_RESERVED_ADDRESS]

  Array of addresses reserved for non-ARP-capable SMBus devices.

12.102.1 Detailed Description

The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform.

Definition at line 48 of file SmbusConfig.h.

12.102.2 Member Data Documentation

12.102.2.1 Enable

```
UINT32 PCH_SMBUS_PREMEM_CONFIG::Enable
```

This member describes whether or not the SMBus controller of PCH should be enabled.

0: Disable; **1: Enable**.

Definition at line 57 of file SmbusConfig.h.

12.102.2.2 Header

```
CONFIG_BLOCK_HEADER PCH_SMBUS_PREMEM_CONFIG::Header
```

Revision 1: Init version.

Config Block Header

Definition at line 52 of file SmbusConfig.h.

12.102.2.3 SpdWriteDisable

```
UINT32 PCH_SMBUS_PREMEM_CONFIG::SpdWriteDisable
```

(Test) SPD Write Disable, 0: leave SPD Write Disable bit; **1: set SPD Write Disable bit.**

For security recommendations, SPD write disable bit must be set.

Definition at line 64 of file SmbusConfig.h.

The documentation for this struct was generated from the following file:

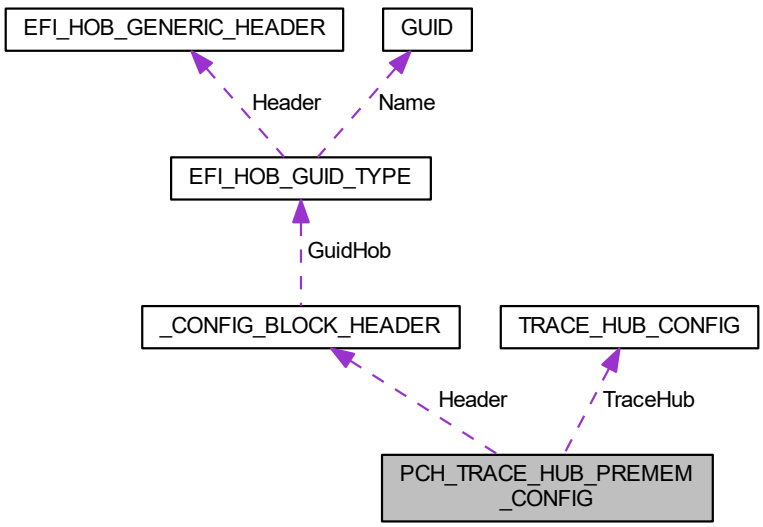
- SmbusConfig.h

12.103 PCH_TRACE_HUB_PREMEM_CONFIG Struct Reference

PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing **Revision 1**: - Initial version.

```
#include <TraceHubConfig.h>
```

Collaboration diagram for PCH_TRACE_HUB_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- TRACE_HUB_CONFIG TraceHub

  Trace Hub Config.

12.103.1 Detailed Description

PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing **Revision 1**: - Initial version.

Definition at line 120 of file TraceHubConfig.h.

The documentation for this struct was generated from the following file:

- TraceHubConfig.h

12.104 PCH_WAKE_CONFIG Struct Reference

This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events.

`#include <PmConfig.h>`

Public Attributes

- UINT32 PmeB0S5Dis: 1

  Corresponds to the PME_B0_S5_DIS bit in the General PM Configuration B (GEN_PMCON_B) register.
- UINT32 WolEnableOverride: 1

  *Corresponds to the "WOL Enable Override" bit in the General PM Configuration B (GEN_PMCON_B) register. 0: Disable; **1: Enable**.*
- UINT32 PcieWakeFromDeepSx: 1

  *Determine if enable PCIe to wake from deep Sx. **0: Disable**; 1: Enable.*
- UINT32 WoWlanEnable: 1

  *Determine if WLAN wake from Sx, corresponds to the "HOST_WLAN_PP_EN" bit in the PWRM_CFG3 register. **0: Disable**; 1: Enable.*
- UINT32 WoWlanDeepSxEnable: 1

  *Determine if WLAN wake from DeepSx, corresponds to the "DSX_WLAN_PP_EN" bit in the PWRM_CFG3 register. **0: Disable**; 1: Enable.*
- UINT32 LanWakeFromDeepSx: 1

  *Determine if enable LAN to wake from deep Sx. 0: Disable; **1: Enable**.*

12.104.1 Detailed Description

This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events.

Definition at line 62 of file PmConfig.h.

12.104.2 Member Data Documentation

12.104.2.1 PmeB0S5Dis

```
UINT32 PCH_WAKE_CONFIG::PmeB0S5Dis
```

Corresponds to the PME_B0_S5_DIS bit in the General PM Configuration B (GEN_PMCON_B) register.

When set to 1, this bit blocks wake events from PME_B0_STS in S5, regardless of the state of PME_B0_EN. When cleared (default), wake events from PME_B0_STS are allowed in S5 if PME_B0_EN = 1. **0: Disable**; 1: Enable.

Definition at line 68 of file PmConfig.h.

The documentation for this struct was generated from the following file:

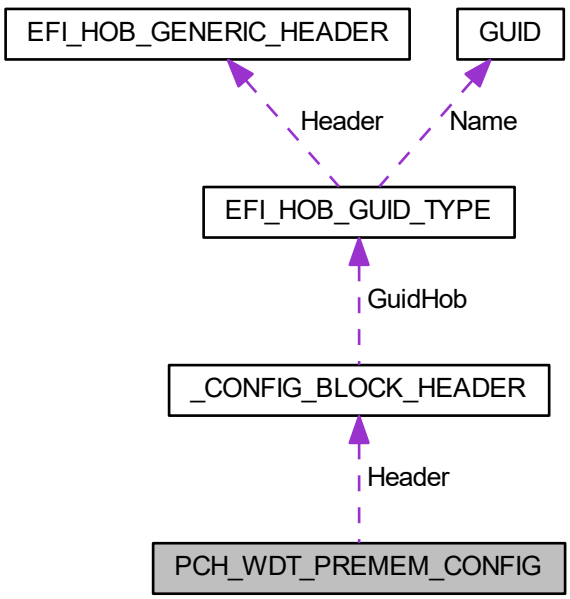
- PmConfig.h

12.105 PCH_WDT_PREMEM_CONFIG Struct Reference

This policy clears status bits and disable watchdog, then lock the WDT registers.

```
#include <WatchDogConfig.h>
```

Collaboration diagram for PCH_WDT_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 DisableAndLock: 1

  ***(Test)** Set 1 to clear WDT status, then disable and lock WDT registers. **0: Disable**; 1: Enable.*

12.105.1 Detailed Description

This policy clears status bits and disable watchdog, then lock the WDT registers.

while WDT is designed to be disabled and locked by Policy, bios should not enable WDT by WDT PPI. In such case, bios shows the warning message but not disable and lock WDT register to make sure WDT event trigger correctly.

Definition at line 51 of file WatchDogConfig.h.

The documentation for this struct was generated from the following file:

- WatchDogConfig.h

12.106 PCIE_COMMON_CONFIG Struct Reference

PCIe Common Config.

```
#include <PcieConfig.h>
```

Public Attributes

- UINT32 RpFunctionSwap: 1

  RpFunctionSwap allows BIOS to use root port function number swapping when root port of function 0 is disabled.
- UINT32 ComplianceTestMode: 1

  Compliance Test Mode shall be enabled when using Compliance Load Board.
- UINT32 RsvdBits0: 29

  Reserved bits.
- UINT8 EnablePort8xhDecode

  ***(Test)** This member describes whether PCIE root port Port 8xh Decode is enabled.*

12.106.1 Detailed Description

PCIe Common Config.

Note

This structure will be expanded to hold all common PCIe policies between SA and PCH

Definition at line 302 of file PcieConfig.h.

12.106.2 Member Data Documentation

12.106.2.1 ComplianceTestMode

`UINT32 PCIE_COMMON_CONFIG::ComplianceTestMode`

Compliance Test Mode shall be enabled when using Compliance Load Board.

0: Disable, 1: Enable

Definition at line 323 of file PcieConfig.h.

12.106.2.2 EnablePort8xhDecode

`UINT8 PCIE_COMMON_CONFIG::EnablePort8xhDecode`

(Test) This member describes whether PCIE root port Port 8xh Decode is enabled.

0: Disable; 1: Enable.

Definition at line 329 of file PcieConfig.h.

12.106.2.3 RpFunctionSwap

`UINT32 PCIE_COMMON_CONFIG::RpFunctionSwap`

RpFunctionSwap allows BIOS to use root port function number swapping when root port of function 0 is disabled.

A PCIE device can have higher functions only when Function0 exists. To satisfy this requirement, BIOS will always enable Function0 of a device that contains more than 0 enabled root ports.

- **Enabled: One of enabled root ports get assigned to Function0.** This offers no guarantee that any particular root port will be available at a specific DevNr:FuncNr location
- Disabled: Root port that corresponds to Function0 will be kept visible even though it might be not used. That way rootport - to - DevNr:FuncNr assignment is constant. This option will impact ports 1, 9, 17. NOTE: This option will not work if ports 1, 9, 17 are fused or configured for RST PCIe storage or disabled through policy In other words, it only affects ports that would become hidden because they have no device connected. NOTE: Disabling function swap may have adverse impact on power management. This option should ONLY be used when each one of root ports 1, 9, 17:
  - is configured as PCIe and has correctly configured ClkReq signal, or
  - does not own any mPhy lanes (they are configured as SATA or USB)

Definition at line 318 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

- PcieConfig.h

12.107 PCIE_DEVICE_OVERRIDE Struct Reference

PCIe device table entry entry.

`#include <PcieConfig.h>`

Public Attributes

- UINT16 VendorId

  The vendor Id of Pci Express card ASPM setting override, 0xFFFF means any Vendor ID.
- UINT16 DeviceId

  The Device Id of Pci Express card ASPM setting override, 0xFFFF means any Device ID.
- UINT8 RevId

  The Rev Id of Pci Express card ASPM setting override, 0xFF means all steppings.
- UINT8 BaseClassCode

  The Base Class Code of Pci Express card ASPM setting override, 0xFF means all base class.
- UINT8 SubClassCode

  The Sub Class Code of Pci Express card ASPM setting override, 0xFF means all sub class.
- UINT8 EndPointAspm

  Override device ASPM (see: PCH_PCIE_ASPM_CONTROL) Bit 1 must be set in OverrideConfig for this field to take effect.
- UINT16 OverrideConfig

  The override config bitmap (see: PCH_PCIE_OVERRIDE_CONFIG).
- UINT16 L1SubstatesCapOffset

  The L1Substates Capability Offset Override.
- UINT8 L1SubstatesCapMask

  L1 Substate Capability Mask.
- UINT8 L1sCommonModeRestoreTime

  L1 Substate Port Common Mode Restore Time Override.
- UINT8 L1sTpowerOnScale

  L1 Substate Port Tpower_on Scale Override.
- UINT8 L1sTpowerOnValue

  L1 Substate Port Tpower_on Value Override.
- UINT16 SnoopLatency

  SnoopLatency bit definition Note: All Reserved bits must be set to 0.
- UINT16 NonSnoopLatency

  NonSnoopLatency bit definition Note: All Reserved bits must be set to 0.
- UINT8 ForceLtrOverride

  Forces LTR override to be permanent The default way LTR override works is: rootport uses LTR override values provided by BIOS until connected device sends an LTR message, then it will use values from the message This settings allows force override of LTR mechanism.

12.107.1 Detailed Description

PCIe device table entry entry.

The PCIe device table is being used to override PCIe device ASPM settings. To take effect table consisting of such entries must be instelled as PPI on gPchPcieDeviceTablePpiGuid. Last entry VendorId must be 0.

Definition at line 342 of file PcieConfig.h.

12.107.2 Member Data Documentation

12.107.2.1 ForceLtrOverride

```
UINT8 PCIE_DEVICE_OVERRIDE::ForceLtrOverride
```

Forces LTR override to be permanent The default way LTR override works is: rootport uses LTR override values provided by BIOS until connected device sends an LTR message, then it will use values from the message This settings allows force override of LTR mechanism.

If it's enabled, then: rootport will use LTR override values provided by BIOS forever; LTR messages sent from connected device will be ignored

Definition at line 436 of file PcieConfig.h.

12.107.2.2 L1sCommonModeRestoreTime

```
UINT8 PCIE_DEVICE_OVERRIDE::L1sCommonModeRestoreTime
```

L1 Substate Port Common Mode Restore Time Override.

(applicable if bit 2 is set in OverrideConfig) L1sCommonModeRestoreTime and L1sTpowerOnScale can have a valid value of 0, but not the L1sTpowerOnValue. If L1sTpowerOnValue is zero, all L1sCommonModeRestoreTime, L1sTpowerOnScale, and L1sTpowerOnValue are ignored, and only L1SubstatesCapOffset is override.

Definition at line 368 of file PcieConfig.h.

12.107.2.3 L1sTpowerOnScale

```
UINT8 PCIE_DEVICE_OVERRIDE::L1sTpowerOnScale
```

L1 Substate Port Tpower_on Scale Override.

(applicable if bit 2 is set in OverrideConfig) L1sCommonModeRestoreTime and L1sTpowerOnScale can have a valid value of 0, but not the L1sTpowerOnValue. If L1sTpowerOnValue is zero, all L1sCommonModeRestoreTime, L1sTpowerOnScale, and L1sTpowerOnValue are ignored, and only L1SubstatesCapOffset is override.

Definition at line 375 of file PcieConfig.h.

12.107.2.4 L1sTpowerOnValue

```
UINT8 PCIE_DEVICE_OVERRIDE::L1sTpowerOnValue
```

L1 Substate Port Tpower_on Value Override.

(applicable if bit 2 is set in OverrideConfig) L1sCommonModeRestoreTime and L1sTpowerOnScale can have a valid value of 0, but not the L1sTpowerOnValue. If L1sTpowerOnValue is zero, all L1sCommonModeRestoreTime, L1sTpowerOnScale, and L1sTpowerOnValue are ignored, and only L1SubstatesCapOffset is override.

Definition at line 382 of file PcieConfig.h.

12.107.2.5 L1SubstatesCapMask

```
UINT8 PCIE_DEVICE_OVERRIDE::L1SubstatesCapMask
```

L1 Substate Capability Mask.

(applicable if bit 2 is set in OverrideConfig) Set to zero then the L1 Substate Capability [3:0] is ignored, and only L1s values are override. Only bit [3:0] are applicable. Other bits are ignored.

Definition at line 361 of file PcieConfig.h.

12.107.2.6 L1SubstatesCapOffset

```
UINT16 PCIE_DEVICE_OVERRIDE::L1SubstatesCapOffset
```

The L1Substates Capability Offset Override.

(applicable if bit 2 is set in OverrideConfig) This field can be zero if only the L1 Substate value is going to be override.

Definition at line 355 of file PcieConfig.h.

12.107.2.7 NonSnoopLatency

```
UINT16 PCIE_DEVICE_OVERRIDE::NonSnoopLatency
```

NonSnoopLatency bit definition Note: All Reserved bits must be set to 0.

BIT[15] - When set to 1b, indicates that the values in bits 9:0 are valid When clear values in bits 9:0 will be ignored BITS[14:13] - Reserved BITS[12:10] - Value in bits 9:0 will be multiplied with the scale in these bits 000b - 1 ns 001b - 32 ns 010b - 1024 ns 011b - 32,768 ns 100b - 1,048,576 ns 101b - 33,554,432 ns 110b - Reserved 111b - Reserved BITS[9:0] - Non Snoop Latency Value. The value in these bits will be multiplied with the scale in bits 12:10

This field takes effect only if bit 3 is set in OverrideConfig.

Definition at line 427 of file PcieConfig.h.

12.107.2.8 SnoopLatency

```
UINT16 PCIE_DEVICE_OVERRIDE::SnoopLatency
```

SnoopLatency bit definition Note: All Reserved bits must be set to 0.

BIT[15] - When set to 1b, indicates that the values in bits 9:0 are valid When clear values in bits 9:0 will be ignored BITS[14:13] - Reserved BITS[12:10] - Value in bits 9:0 will be multiplied with the scale in these bits 000b - 1 ns 001b - 32 ns 010b - 1024 ns 011b - 32,768 ns 100b - 1,048,576 ns 101b - 33,554,432 ns 110b - Reserved 111b - Reserved BITS[9:0] - Snoop Latency Value. The value in these bits will be multiplied with the scale in bits 12:10

This field takes effect only if bit 3 is set in OverrideConfig.

Definition at line 405 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

- PcieConfig.h

12.108 PCIE_EQ_PARAM Struct Reference

Represent lane specific PCIe Gen3 equalization parameters.

```
#include <PcieConfig.h>
```

Public Attributes

- UINT8 Cm
  Coefficient C-1.
- UINT8 Cp
  Coefficient C+1.
- UINT8 Rsvd0 [2]
  Reserved bytes.

12.108.1 Detailed Description

Represent lane specific PCIe Gen3 equalization parameters.

Definition at line 74 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

- PcieConfig.h

12.109 PCIE_IMR_CONFIG Struct Reference

PCIe IMR Config.

```
#include <PciePreMemConfig.h>
```

Public Attributes

- UINT8 ImrEnabled

  *PCIe IMR. **0: Disable**; 1: Enable.*
- UINT8 ImrRpLocation

  0: PCH_PCIe; 1: CPU_PCIe. If PCIeImrEnabled is TRUE then this will use to select the Root port location from PCH PCIe or CPU PCIe.Refer PCIE_IMR_ROOT_PORT_LOCATION above
- UINT16 ImrSize

  PCIe IMR size in megabytes.
- UINT8 ImrRpSelection

  Index of root port that is selected for PCIe IMR (0 based)

12.109.1 Detailed Description

PCIe IMR Config.

Definition at line 53 of file PciePreMemConfig.h.

The documentation for this struct was generated from the following file:

- PciePreMemConfig.h

12.110 PCIE_LINK_EQ_PLATFORM_SETTINGS Struct Reference

PCIe Link EQ Platform Settings.

```
#include <PcieConfig.h>
```

Public Attributes

- UINT8 PcieLinkEqMethod

  Tells BIOS which link EQ method should be used for this port. Please refer to PCIE_LINK_EQ_METHOD for details of supported methods. Default: PcieLinkHardwareEq.
- UINT8 PcieLinkEqMode

  Tells BIOS which mode should be used for PCIe link EQ. Please refer to PCIE_LINK_EQ_MODE for details of supported modes. Default: depends on SoC.
- UINT8 LocalTxOverrideEn

  Specifies if BIOS should perform local transmitter override during phase 2 of EQ process.
- UINT8 DefaultPresetEnable [PCIE_LINK_EQ_PRESETS_MAX]

  Specifies if default settings should remain in use.
- UINT8 HapcsSearchEnable

  Specifies setting for Hardware Autonomous Preset/Coefficient Search mechanism.
- UINT32 Ph2LocalTxOverridePreset

  Specifies the preset that should be used during local transmitter override during phase 2 of EQ process.
- UINT32 PCETTimer

  PCET Timer value for single PCIe speed.
- UINT8 RemotePresetCoeffoverride

  Remote Transmitter Preset Coefficient Override for single PCIe speed.
- UINT8 EqPh3Bypass

PCIe Equalization Phase 3 Enable Control.

- UINT8 EqPh23Bypass

  PCIe Equalization Phase 2-3 Enable Control.

- UINT8 TsLockTimer

  8.0GT/s Training Sequence Wait Latency For Presets / Coefficients Evaluation - Gen3 TS Lock Timer

- UINT8 EqPhBypass

  PCIe Equalization Phase Enable Control.

- UINT8 Ph3NoOfPresetOrCoeff

  Tells BIOS how many presets/coefficients should be used during link EQ.

- PCIE_LINK_EQ_COEFFICIENTS Ph3CoefficientsList [PCIE_LINK_EQ_COEFFICIENTS_MAX]

  List of the PCIe coefficients to be used during equalization process. Only valid if PcieLinkEqMode is PcieLinkEq↩ CoefficientMode.

- UINT32 Ph3PresetList [PCIE_LINK_EQ_PRESETS_MAX]

  List of the PCIe preset values to be used during equalization process. Only valid if PcieLinkEqMode is PcieLinkEq↩ PresetMode.

- UINT32 EndPointPresetList [PCIE_LINK_EQ_PRESETS_MAX]

  List of the PCIe preset values to be used during equalization process. Only valid if PcieLinkEqMode is PcieLinkEq↩ PresetMode.

- UINT32 HintList [PCIE_LINK_EQ_PRESETS_MAX]

  List of the Hint values to be used during equalization process for preset calculation.

- UINT32 Ph1DpTxPreset

  Specifies the value of the downstream port transmitter preset to be used during phase 1 of the equalization process. Will be applied to all lanes.

- UINT32 Ph1UpTxPreset

  Specifies the value of the upstream port transmitter preset to be used during phase 1 of the equalization process. Will be applied to all lanes.

12.110.1 Detailed Description

PCIe Link EQ Platform Settings.

Definition at line 131 of file PcieConfig.h.

12.110.2 Member Data Documentation

12.110.2.1 EqPh23Bypass

```
UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::EqPh23Bypass
```

PCIe Equalization Phase 2-3 Enable Control.

- **Disabled** (0x0) : Disable Phase 2 - Phase 3 (Default)
- Enabled (0x1) : Enable Phase 2 - Phase 3

Definition at line 162 of file PcieConfig.h.

12.110.2.2 EqPh3Bypass

`UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::EqPh3Bypass`

PCIe Equalization Phase 3 Enable Control.

- **Disabled** (0x0) : Disable phase 3 (Default)
  - Enabled (0x1) : Enable phase 3

Definition at line 156 of file PcieConfig.h.

12.110.2.3 EqPhBypass

`UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::EqPhBypass`

PCIe Equalization Phase Enable Control.

- **Disabled** (0x0) : Disable Phase (Default)
- Enabled (0x1) : Enable Phase

Definition at line 169 of file PcieConfig.h.

12.110.2.4 LocalTxOverrideEn

`UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::LocalTxOverrideEn`

Specifies if BIOS should perform local transmitter override during phase 2 of EQ process.

If enabled value in Ph2LocalTxOverridePreset must be valid. **0: Disabled**; 1: Enabled

Definition at line 139 of file PcieConfig.h.

12.110.2.5 Ph2LocalTxOverridePreset

`UINT32 PCIE_LINK_EQ_PLATFORM_SETTINGS::Ph2LocalTxOverridePreset`

Specifies the preset that should be used during local transmitter override during phase 2 of EQ process.

Used only if LocalTxOverrideEn is TRUE. Will be applied to all PCIe lanes of the root port. Valid up to the PCIE_↩ LINK_EQ_PRESET_MAX value. **Default: 0**< >

Definition at line 148 of file PcieConfig.h.

12.110.2.6 Ph3NoOfPresetOrCoeff

```
UINT8 PCIE_LINK_EQ_PLATFORM_SETTINGS::Ph3NoOfPresetOrCoeff
```

Tells BIOS how many presets/coefficients should be used during link EQ.

Entries in the Ph3CoefficientsList or Ph3PresetList(depending on chosen mode) need to be valid up to the number specified in this field.

Definition at line 175 of file PcieConfig.h.

The documentation for this struct was generated from the following file:

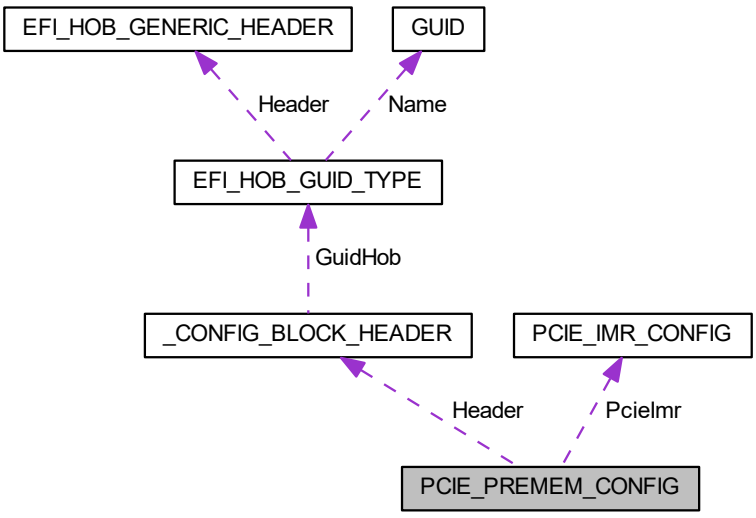
- PcieConfig.h

12.111 PCIE_PREMEM_CONFIG Struct Reference

PCIe Pre-Memory Configuration **Revision 1**: - Initial version.

```
#include <PciePreMemConfig.h>
```

Collaboration diagram for PCIE_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0 - 27 Config Block Header.
- PCIE_IMR_CONFIG PcieImr

  IMR Configuration.

12.111.1 Detailed Description

PCIe Pre-Memory Configuration **Revision 1**: - Initial version.

Definition at line 65 of file PciePreMemConfig.h.

The documentation for this struct was generated from the following file:

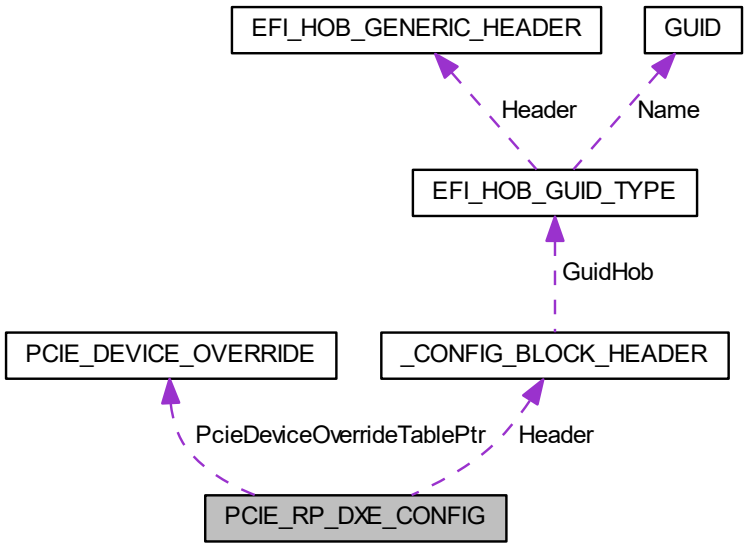
- PciePreMemConfig.h

12.112 PCIE_RP_DXE_CONFIG Struct Reference

The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase.

```
#include <PchPcieRpConfig.h>
```

Collaboration diagram for PCIE_RP_DXE_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- PCIE_DEVICE_OVERRIDE * PcieDeviceOverrideTablePtr

  PCIe device override table The PCIe device table is being used to override PCIe device ASPM settings.

12.112.1 Detailed Description

The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase.

Revision 1:

- Init version

Definition at line 290 of file PchPcieRpConfig.h.

12.112.2 Member Data Documentation

12.112.2.1 PcieDeviceOverrideTablePtr

```
PCIE_DEVICE_OVERRIDE* PCIE_RP_DXE_CONFIG::PcieDeviceOverrideTablePtr
```

PCIe device override table The PCIe device table is being used to override PCIe device ASPM settings.

And it's only used in DXE phase. Please refer to PCIE_DEVICE_OVERRIDE structure for the table. Last entry VendorId must be 0.

Definition at line 300 of file PchPcieRpConfig.h.

The documentation for this struct was generated from the following file:

- PchPcieRpConfig.h

12.113 PMC_GLOBAL_RESET_MASK Union Reference

Description of Global Reset Trigger/Event Mask register.

```
#include <PmConfig.h>
```

12.113.1 Detailed Description

Description of Global Reset Trigger/Event Mask register.

Definition at line 163 of file PmConfig.h.

The documentation for this union was generated from the following file:

- PmConfig.h

12.114 PMC_INTERFACE_CONFIG Struct Reference

The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC.

`#include <TcssPeiConfig.h>`

Public Attributes

- UINT8 PmcPdEnable

  PMC PD Solution Enable.

12.114.1 Detailed Description

The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC.

Definition at line 81 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiConfig.h

12.115 PMC_LPM_S0IX_SUB_STATE_EN Union Reference

Low Power Mode Enable config.

`#include <PmConfig.h>`

12.115.1 Detailed Description

Low Power Mode Enable config.

Used to configure if respective S0i2/3 sub-states are to be supported by the platform. Each bit corresponds to one LPM state - LPMx->BITx. Some sub-states will require external FETs controlled by EXT_PWR_GATE#/EXT_P↩ WR_GATE2# pins to gate v1p05-PHY or v1p05-IS supplies

Definition at line 130 of file PmConfig.h.

12.115.2 Member Data Documentation

12.115.2.1 S0i2p2En

```
UINT32 PMC_LPM_S0IX_SUB_STATE_EN::S0i2p2En
```

LPM2 - S0i2.2 Enable.

Requires EXT_PWR_GATE# controlled FET to gate v1p05 PHY. Refer to V1p05PhyExtFetControlEn.

Definition at line 139 of file PmConfig.h.

12.115.2.2 S0i3p3En

```
UINT32 PMC_LPM_S0IX_SUB_STATE_EN::S0i3p3En
```

LPM5 - S0i3.3 Enable.

Requires EXT_PWR_GATE# controlled FET to gate v1p05 PHY. Refer to V1p05PhyExtFetControlEn.

Definition at line 148 of file PmConfig.h.

12.115.2.3 S0i3p4En

```
UINT32 PMC_LPM_S0IX_SUB_STATE_EN::S0i3p4En
```

LPM7 - S0i3.4 Enable.

Requires EXT_PWR_GATE2# controlled FET to gate v1p05-SRAM/ISCLK. Refer to V1p05IsExtFetControlEn.

Definition at line 154 of file PmConfig.h.

The documentation for this union was generated from the following file:

- PmConfig.h

12.116 PROTECTED_RANGE Struct Reference

Protected Flash Range.

```
#include <FlashProtectionConfig.h>
```

Public Attributes

- UINT32 WriteProtectionEnable: 1

  *Write or erase is blocked by hardware. **0: Disable**; 1: Enable.*
- UINT32 ReadProtectionEnable: 1

  *Read is blocked by hardware. **0: Disable**; 1: Enable.*
- UINT32 RsvdBits: 30

  Reserved.
- UINT16 ProtectedRangeLimit

  The address of the upper limit of protection This is a left shifted address by 12 bits with address bits 11:0 are assumed to be FFFh for limit comparison.
- UINT16 ProtectedRangeBase

  The address of the upper limit of protection This is a left shifted address by 12 bits with address bits 11:0 are assumed to be 0.

12.116.1 Detailed Description

Protected Flash Range.

Definition at line 51 of file FlashProtectionConfig.h.

The documentation for this struct was generated from the following file:

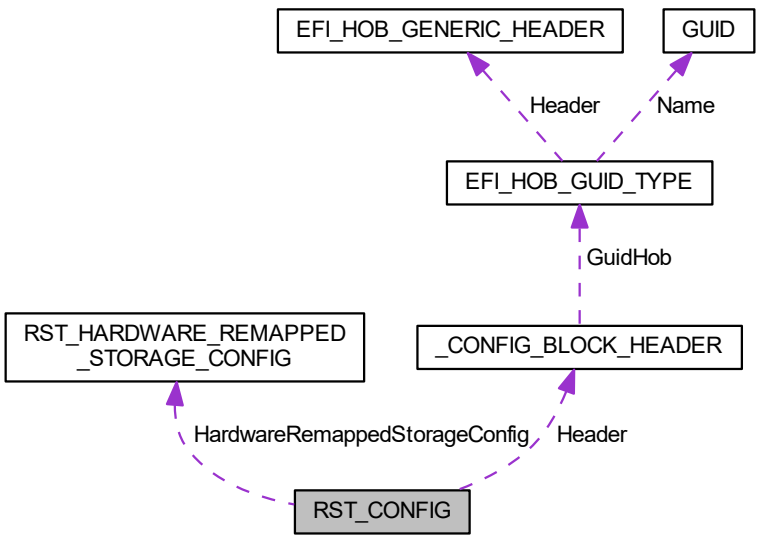
- FlashProtectionConfig.h

12.117 RST_CONFIG Struct Reference

Rapid Storage Technology settings.

`#include <RstConfig.h>`

Collaboration diagram for RST_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 Raid0: 1

  0 : Disable; ***1 : Enable*** *RAID0*
- UINT32 Raid1: 1

  0 : Disable; ***1 : Enable*** *RAID1*
- UINT32 Raid10: 1

  0 : Disable; ***1 : Enable*** *RAID10*
- UINT32 Raid5: 1

  0 : Disable; ***1 : Enable*** *RAID5*
- UINT32 Irrt: 1

  0 : Disable; ***1 : Enable*** *Intel Rapid Recovery Technology*
- UINT32 OromUiBanner: 1

  0 : Disable; ***1 : Enable*** *OROM UI and BANNER*
- UINT32 OromUiDelay: 2

  00b : 2 secs*; 01b : 4 secs; 10b : 6 secs; 11 : 8 secs (see : SATA_OROM_DELAY)*
- UINT32 HddUnlock: 1

  0 : Disable; ***1 : Enable****. Indicates that the HDD password unlock in the OS is enabled*
- UINT32 LedLocate: 1

  0 : Disable; ***1 : Enable****. Indicates that the LED/SGPIO hardware is attached and ping to locate feature is enabled on the OS*
- UINT32 IrrtOnly: 1

  0 : Disable; ***1 : Enable****. Allow only IRRT drives to span internal and external ports*
- UINT32 SmartStorage: 1

  0 : Disable; ***1 : Enable*** *RST Smart Storage caching Bit*
- UINT32 LegacyOrom: 1

  0 : Disable*; 1 : Enable RST Legacy OROM*
- UINT32 OptaneMemory: 1

  0: Disable; ***1: Enable*** *RST Optane(TM) Memory*
- UINT32 CpuAttachedStorage: 1

  0: Disable; ***1: Enable*** *CPU Attached Storage*
- UINT32 RsvdBits0: 17

  Reserved Bits.
- RST_HARDWARE_REMAPPED_STORAGE_CONFIG HardwareRemappedStorageConfig [PCH_MAX_↩ RST_PCIE_STORAGE_CR]

  This member describes the details of implementation of Intel RST for PCIe Storage remapping (Intel RST Driver is required) Note: RST for PCIe Sorage remapping is supported only for first SATA controller if more controllers are available.

12.117.1 Detailed Description

Rapid Storage Technology settings.

Revision 1:

- Initial version.

Definition at line 84 of file RstConfig.h.

The documentation for this struct was generated from the following file:

- RstConfig.h

12.118 RST_HARDWARE_REMAPPED_STORAGE_CONFIG Struct Reference

This structure describes the details of Intel RST for PCIe Storage remapping Note: In order to use this feature, Intel RST Driver is required.

```
#include <RstConfig.h>
```

Public Attributes

- UINT32 Enable: 1

  This member describes whether or not the Intel RST for PCIe Storage remapping should be enabled.
- UINT32 RstPcieStoragePort: 5

  *Intel RST for PCIe Storage remapping - PCIe Port Selection (1-based, **0 = autodetect**) The supported ports for PCIe Storage remapping is different depend on the platform and cycle router.*
- UINT32 DeviceResetDelay: 8

  PCIe Storage Device Reset Delay in milliseconds (ms), which it guarantees such delay gap is fulfilled before PCIe Storage Device configuration space is accessed after an reset caused by the link disable and enable step.
- UINT32 RsvdBits0: 18

  Reserved bits.

12.118.1 Detailed Description

This structure describes the details of Intel RST for PCIe Storage remapping Note: In order to use this feature, Intel RST Driver is required.

Definition at line 56 of file RstConfig.h.

12.118.2 Member Data Documentation

12.118.2.1 DeviceResetDelay

```
UINT32 RST_HARDWARE_REMAPPED_STORAGE_CONFIG::DeviceResetDelay
```

PCIe Storage Device Reset Delay in milliseconds (ms), which it guarantees such delay gap is fulfilled before PCIe Storage Device configuration space is accessed after an reset caused by the link disable and enable step.

Default value is **100ms**.

Definition at line 73 of file RstConfig.h.

12.118.2.2 Enable

```
UINT32 RST_HARDWARE_REMAPPED_STORAGE_CONFIG::Enable
```

This member describes whether or not the Intel RST for PCIe Storage remapping should be enabled.

0: Disable; 1: Enable. Note 1: If Sata Controller is disabled, PCIe Storage Remapping should be disabled as well Note 2: If PCIe Storage remapping is enabled, the PCH integrated AHCI controllers Class Code is configured as RAID

Definition at line 62 of file RstConfig.h.

The documentation for this struct was generated from the following file:

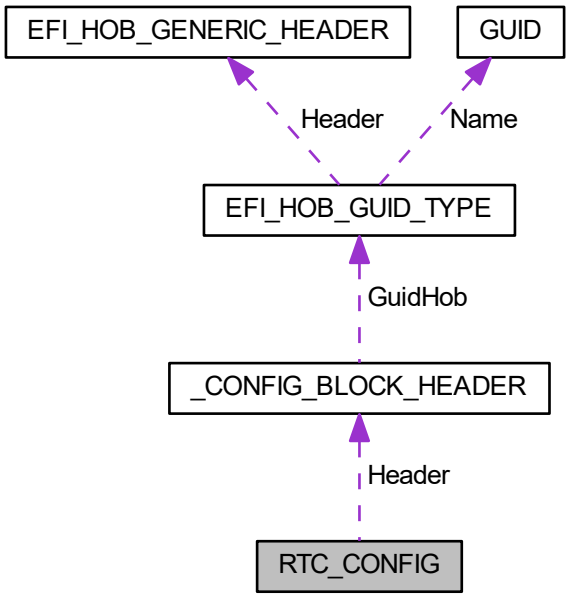
- RstConfig.h

12.119 RTC_CONFIG Struct Reference

The RTC_CONFIG block describes the expected configuration of RTC configuration.

```
#include <RtcConfig.h>
```

Collaboration diagram for RTC_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 BiosInterfaceLock: 1

  When set, prevents RTC TS (BUC.TS) from being changed.
- UINT32 MemoryLock: 1

  When set, bytes 38h-3Fh in the upper 128bytes bank of RTC RAM are locked and cannot be accessed.

12.119.1 Detailed Description

The RTC_CONFIG block describes the expected configuration of RTC configuration.

Definition at line 48 of file RtcConfig.h.

12.119.2 Member Data Documentation

12.119.2.1 BiosInterfaceLock

```
UINT32 RTC_CONFIG::BiosInterfaceLock
```

When set, prevents RTC TS (BUC.TS) from being changed.

This BILD bit has different function compared to LPC/eSPI, SPI. 0: Disabled; **1: Enabled**

Definition at line 55 of file RtcConfig.h.

12.119.2.2 MemoryLock

```
UINT32 RTC_CONFIG::MemoryLock
```

When set, bytes 38h-3Fh in the upper 128bytes bank of RTC RAM are locked and cannot be accessed.

Writes will be droipped and reads will not return any guaranteed data. 0: Disabled; **1: Enabled**

Definition at line 62 of file RtcConfig.h.

The documentation for this struct was generated from the following file:

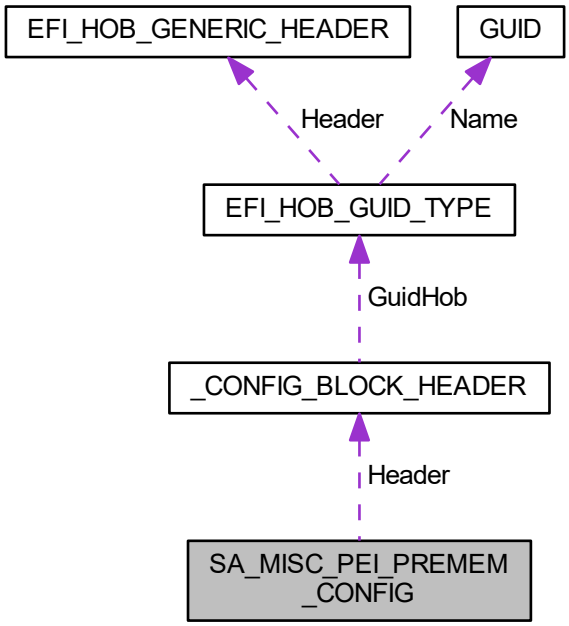
- RtcConfig.h

12.120 SA_MISC_PEI_PREMEM_CONFIG Struct Reference

This configuration block is to configure SA Miscellaneous variables during PEI Pre-Mem phase like programming different System Agent BARs, TsegSize, MmioSize required etc.

`#include <SaMiscPeiPreMemConfig.h>`

Collaboration diagram for SA_MISC_PEI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 SpdAddressTable [MEM_CFG_MAX_SOCKETS]

  Offset 28 Memory DIMMs' SPD address for reading SPD data.
- VOID * S3DataPtr

  Offset 44 Memory data save pointer for S3 resume. The memory space should be allocated and filled with proper S3 resume data on a resume path.
- UINT32 SmbusBar

  Offset 48 Address of System Agent SMBUS BAR: ***0xEFA0***
- UINT32 TsegSize

  Offset 52 Size of TSEG in bytes.
- UINT32 SkipExtGfxScan:1

  (Test) *Offset 56:0 :1=Skip External Gfx Device Scan;* ***0=Scan for external graphics devices****. Set this policy to skip External Graphics card scanning if the platform uses Internal Graphics only.*
- UINT32 BdatEnable:1

*Offset 56:1 :This field enables the generation of the BIOS DATA ACPI Tables: **0=FALSE**, 1=TRUE.*

- UINT32 TxtImplemented:1

  *OFfset 56:2 :This field currently is used to tell MRC if it should run after TXT initializatoin completed: **0=Run without waiting for TXT**, 1=Run after TXT initialization by callback.*

- UINT32 ScanExtGfxForLegacyOpRom:1

  *Offset 56:3 : **(Test)** Scan External Discrete Graphics Devices for Legacy Only VGA OpROMs.*

- UINT32 RsvdBits0:28

  Offset 56:4 :Reserved for future use.

- UINT8 UserBd

  *Offset 60 **0=Mobile/Mobile Halo**, 1=Desktop/DT Halo, 5=ULT/ULX/Mobile Halo, 7=UP Server.*

- UINT8 LockPTMregs

  (Test)** Offset 61 Lock PCU Thermal Management registers: 0=FALSE, **1=TRUE

- UINT8 BdatTestType

  *Offset 62 When BdatEnable is set to TRUE, this option selects the type of data which will be populated in the BIOS Data ACPI Tables: **0=RMT**, 1=RMT Per Bit, 2=Margin 2D.*

- UINT8 Rsvd4

  Offset 63 Reserved for config block alignment.

- UINT32 AcpiReservedMemorySize

  Offset 64 The Size of a Reserved memory buffer allocated in previous boot for S3 resume used. Originally it is retrieved from AcpiVariableCompatibility variable.

- UINT32 OpRomScanTempMmioBar

  ***(Test)** Offset 68 Temporary address to MMIO map OpROMs during VGA scanning. Used for ScanExtGfxForLegacy↩ OpRom feature. MUST BE 16MB ALIGNED!*

- UINT32 OpRomScanTempMmioLimit

  ***(Test)** Offset 72 Limit address for OpROM MMIO range. Used for ScanExtGfxForLegacyOpRom feature. (OpROM↩ ScanTempMmioLimit - OpRomScanTempMmioBar) MUST BE >= 16MB!*

- UINT32 Rsvd3

  Offset 76 Reserved for config block alignment.

- UINT64 AcpiReservedMemoryBase

  Offset 80 The Base address of a Reserved memory buffer allocated in previous boot for S3 resume used. Originally it is retrieved from AcpiVariableCompatibility variable.

- UINT64 SystemMemoryLength

  Offset 88 Total system memory length from previous boot, this is required for S3 resume. Originally it is retrieved from AcpiVariableCompatibility variable.

- UINT8 DisableMrcRetrainingOnRtcPowerLoss

  Offset 96 Enable/Disable DisableMrcRetrainingOnRtcPowerLoss.

- UINT8 Reserved1 [3]

  Offset 97 Reserved for config block alignment.

- UINT8 OfflineCrashDumpEnable

  Offset 100 OfflineCrashDump feature enable. If this is enabled, memory must be reserved for BIOS execution for when an OfflineCrashDump occurs.

- UINT8 Rsvd [4]

  Reserved for config block alignment.

- UINT8 CkdAddressTable [MEM_CFG_MAX_SOCKETS]

  Memory DIMMs' CKD address for read/write CKD data.

- UINT16 MrcMustStaticSpdData

  Although BIOS provides spd address, MRC need use static spd data in VPD.

12.120.1 Detailed Description

This configuration block is to configure SA Miscellaneous variables during PEI Pre-Mem phase like programming different System Agent BARs, TsegSize, MmioSize required etc.

Revision 1:

- Initial version. **Revision 2**:
- Add CkdAddress Table. **Revision 3**:
- Add MrcMustStaticSpdData.

Definition at line 56 of file SaMiscPeiPreMemConfig.h.

12.120.2 Member Data Documentation

12.120.2.1 CkdAddressTable

```
UINT8 SA_MISC_PEI_PREMEM_CONFIG::CkdAddressTable[MEM_CFG_MAX_SOCKETS]
```

Memory DIMMs' CKD address for read/write CKD data.

0 - Controller 0 Channel 0 Dimm 0 - DDR4 - DDR5 - LPDDR4 - LPDDR5 1 - Controller 0 Channel 0 Dimm 1 - DDR4 2 - Controller 0 Channel 1 Dimm 0 ------— DDR5 - LPDDR4 - LPDDR5 3 - Controller 0 Channel 1 Dimm 1 ------— DDR5 2DPC 4 - Controller 0 Channel 2 Dimm 0 ----------— LPDDR4 - LPDDR5 6 - Controller 0 Channel 3 Dimm 0 ----------— LPDDR4 - LPDDR5 8 - Controller 1 Channel 0 Dimm 0 - DDR4 - DDR5 - LPDDR4 - LPDDR5 9 - Controller 1 Channel 0 Dimm 1 - DDR4 10 - Controller 1 Channel 1 Dimm 0 ------— DDR5 - LPDDR4 - LPDDR5 11 - Controller 1 Channel 1 Dimm 1 ------— DDR5 2DPC 12 - Controller 1 Channel 2 Dimm 0 ----------— LPDDR4 - LPDDR5 14 - Controller 1 Channel 3 Dimm 0 ----------— LPDDR4 - LPDDR5

Definition at line 142 of file SaMiscPeiPreMemConfig.h.

12.120.2.2 MrcMustStaticSpdData

```
UINT16 SA_MISC_PEI_PREMEM_CONFIG::MrcMustStaticSpdData
```

Although BIOS provides spd address, MRC need use static spd data in VPD.

Each bit maps to one dimm.

Definition at line 148 of file SaMiscPeiPreMemConfig.h.

12.120.2.3 ScanExtGfxForLegacyOpRom

```
UINT32 SA_MISC_PEI_PREMEM_CONFIG::ScanExtGfxForLegacyOpRom
```

Offset 56:3 : **(Test)** Scan External Discrete Graphics Devices for Legacy Only VGA OpROMs.

When enabled, if the primary graphics device is an external discrete graphics device, Si will scan the graphics device for legacy only VGA OpROMs.

This is intended to ease the implementation of a BIOS feature to automatically enable CSM if the Primary Gfx device only supports Legacy VBIOS (No UEFI GOP Present). Otherwise disabling CSM won't result in no video being displayed. This is useful for platforms that implement PCIe slots that allow the end user to install an arbitrary Gfx device.

This setting will only take effect if SkipExtGfxScan == 0. It is ignored otherwise.

- Disabled (0x0) : Don't Scan for Legacy Only VGA OpROMs (Default)
- **Enabled** (0x1) : Scan External Gfx for Legacy Only VGA OpROM

Definition at line 102 of file SaMiscPeiPreMemConfig.h.

12.120.2.4 SpdAddressTable

```
UINT8 SA_MISC_PEI_PREMEM_CONFIG::SpdAddressTable[MEM_CFG_MAX_SOCKETS]
```

Offset 28 Memory DIMMs' SPD address for reading SPD data.

TGL Mapping 0 - Controller 0 Channel 0 Dimm 0 - DDR4 - DDR5 - LPDDR4 - LPDDR5 1 - Controller 0 Channel 0 Dimm 1 - DDR4 2 - Controller 0 Channel 1 Dimm 0 ------— DDR5 - LPDDR4 - LPDDR5 3 - Controller 0 Channel 1 Dimm 1 ------— DDR5 2DPC 4 - Controller 0 Channel 2 Dimm 0 ----------— LPDDR4 - LPDDR5 6 - Controller 0 Channel 3 Dimm 0 ----------— LPDDR4 - LPDDR5 8 - Controller 1 Channel 0 Dimm 0 - DDR4 - DDR5 - LPDDR4 - LPDDR5 9 - Controller 1 Channel 0 Dimm 1 - DDR4 10 - Controller 1 Channel 1 Dimm 0 ------— DDR5 - LPDDR4 - LPDDR5 11 - Controller 1 Channel 1 Dimm 1 ------— DDR5 2DPC 12 - Controller 1 Channel 2 Dimm 0 ----------— LPDDR4 - LPDDR5 14 - Controller 1 Channel 3 Dimm 0 ----------— LPDDR4 - LPDDR5

Definition at line 74 of file SaMiscPeiPreMemConfig.h.

12.120.2.5 TsegSize

```
UINT32 SA_MISC_PEI_PREMEM_CONFIG::TsegSize
```

Offset 52 Size of TSEG in bytes.

(Must be power of 2) **0x400000**: 4MB for Release build 0x1000000 : 16MB for Debug build

Definition at line 82 of file SaMiscPeiPreMemConfig.h.

The documentation for this struct was generated from the following file:

- SaMiscPeiPreMemConfig.h

12.121 SA_XDCI_IRQ_INT_CONFIG Struct Reference

The SA XDCI INT Pin and IRQ number.

`#include <TcssPeiConfig.h>`

Public Attributes

- UINT8 IntPing

  Int Pin Number.
- UINT8 Irq

  Irq Number.

12.121.1 Detailed Description

The SA XDCI INT Pin and IRQ number.

Definition at line 89 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

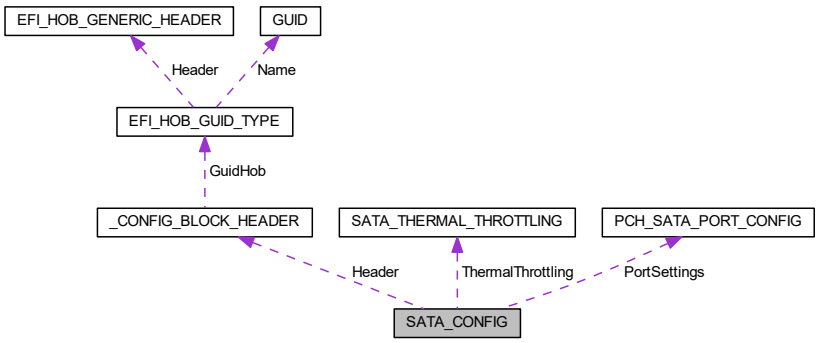
- TcssPeiConfig.h

12.122 SATA_CONFIG Struct Reference

The SATA_CONFIG block describes the expected configuration of the SATA controllers.

`#include <SataConfig.h>`

Collaboration diagram for SATA_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT8 Enable

  This member describes whether or not the SATA controllers should be enabled.
- UINT8 TestMode

  ***(Test) 0: Disable**; 1: Allow entrance to the PCH SATA test modes*
- UINT8 SalpSupport

  *0: Disable; **1: Enable** Aggressive Link Power Management*
- UINT8 PwrOptEnable

  *0: Disable; **1: Enable** SATA Power Optimizer on PCH side.*
- UINT8 EsataSpeedLimit

  EsataSpeedLimit When enabled, BIOS will configure the PxSCTL.SPD to 2 to limit the eSATA port speed.
- UINT8 LedEnable

  *SATA LED indicates SATA controller activity. 0: Disable; **1: Enable** SATA LED.*
- UINT8 RaidDeviceId

  This option allows to configure SATA controller device ID while in RAID mode.
- UINT8 SataRstInterrupt

  *Controlls which interrupts will be linked to SATA controller CAP list This option will take effect only if SATA controller is in RAID mode Default: **PchSataMsix***
- UINT8 SataMode

  Determines the system will be configured to which SATA mode.
- UINT8 SpeedLimit

  Indicates the maximum speed the SATA controller can support.
- UINT8 EnclosureSupport

  Enclosure Management Support. 0: Disable; 1: Enable.
- UINT8 SgpioSupport

  *Controlls whenever Serial GPIO support is enabled for controller **0: Disable**; 1: Enable.*
- PCH_SATA_PORT_CONFIG PortSettings [PCH_MAX_SATA_PORTS]

  This member configures the features, property, and capability for each SATA port.
- SATA_THERMAL_THROTTLING ThermalThrottling

  This field decides the settings of Sata thermal throttling.

12.122.1 Detailed Description

The SATA_CONFIG block describes the expected configuration of the SATA controllers.

Revision 1:

- Initial version. **Revision 2**:
- Added DevSlpGpioPinMux.

Definition at line 133 of file SataConfig.h.

12.122.2 Member Data Documentation

12.122.2.1 Enable

```
UINT8 SATA_CONFIG::Enable
```

This member describes whether or not the SATA controllers should be enabled.

0: Disable; **1: Enable**.

Definition at line 138 of file SataConfig.h.

12.122.2.2 EsataSpeedLimit

```
UINT8 SATA_CONFIG::EsataSpeedLimit
```

EsataSpeedLimit When enabled, BIOS will configure the PxSCTL.SPD to 2 to limit the eSATA port speed.

Please be noted, this setting could be cleared by HBA reset, which might be issued by EFI AHCI driver when POST time, or by SATA inbox driver/RST driver after POST. To support the Speed Limitation when POST, the EFI AHCI driver should preserve the setting before and after initialization. For support it after POST, it's dependent on driver's behavior. **0: Disable**; 1: Enable

Definition at line 152 of file SataConfig.h.

12.122.2.3 RaidDeviceId

```
UINT8 SATA_CONFIG::RaidDeviceId
```

This option allows to configure SATA controller device ID while in RAID mode.

Refer to SATA_RAID_DEV_ID enumeration for supported options. Choosing Client will allow RST driver loading, RSTe driver will not be able to load Choosing Alternate will not allow RST inbox driver loading in Windows Choosing Server will allow RSTe driver loading, RST driver will not load **0: Client**; 1: Alternate; 2: Server

Definition at line 162 of file SataConfig.h.

12.122.2.4 SataMode

```
UINT8 SATA_CONFIG::SataMode
```

Determines the system will be configured to which SATA mode.

Refer to SATA_MODE enumeration for supported options. Default is **SataModeAhci**.

Definition at line 174 of file SataConfig.h.

12.122.2.5 SpeedLimit

```
UINT8 SATA_CONFIG::SpeedLimit
```

Indicates the maximum speed the SATA controller can support.

Refer to SATA_SPEED enumeration for supported options. **0h: SataSpeedDefault**; 1h: 1.5 Gb/s (Gen 1); 2h: 3 Gb/s(Gen 2); 3h: 6 Gb/s (Gen 1)

Definition at line 180 of file SataConfig.h.

12.122.2.6 ThermalThrottling

```
SATA_THERMAL_THROTTLING SATA_CONFIG::ThermalThrottling
```

This field decides the settings of Sata thermal throttling.

When the Suggested Setting is enabled, PCH RC will use the suggested representative values.

Definition at line 195 of file SataConfig.h.

The documentation for this struct was generated from the following file:

- SataConfig.h

12.123 SATA_THERMAL_THROTTLING Struct Reference

This structure lists PCH supported SATA thermal throttling register setting for customization.

```
#include <SataConfig.h>
```

Public Attributes

- UINT32 P0T1M: 2
  Port 0 T1 Multipler.
- UINT32 P0T2M: 2
  Port 0 T2 Multipler.
- UINT32 P0T3M: 2
  Port 0 T3 Multipler.
- UINT32 P0TDisp: 2
  Port 0 Tdispatch.
- UINT32 P1T1M: 2
  Port 1 T1 Multipler.
- UINT32 P1T2M: 2
  Port 1 T2 Multipler.
- UINT32 P1T3M: 2
  Port 1 T3 Multipler.
- UINT32 P1TDisp: 2

Port 1 Tdispatch.

- UINT32 P0Tinact: 2

  Port 0 Tinactive.

- UINT32 P0TDispFinit: 1

  Port 0 Alternate Fast Init Tdispatch.

- UINT32 P1Tinact: 2

  Port 1 Tinactive.

- UINT32 P1TDispFinit: 1

  Port 1 Alternate Fast Init Tdispatch.

- UINT32 SuggestedSetting: 1

  *0: Disable; **1: Enable** suggested representative values*

- UINT32 RsvdBits0: 9

  Reserved bits.

12.123.1 Detailed Description

This structure lists PCH supported SATA thermal throttling register setting for customization.

The settings is programmed through SATA Index/Data registers. When the SuggestedSetting is enabled, the customized values are ignored.

Definition at line 106 of file SataConfig.h.

The documentation for this struct was generated from the following file:

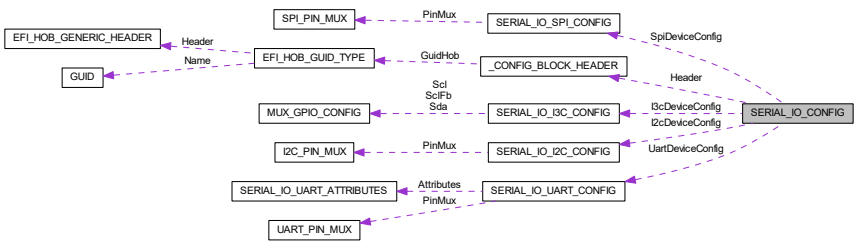
- SataConfig.h

12.124 SERIAL_IO_CONFIG Struct Reference

The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers.

```
#include <SerialIoConfig.h>
```

Collaboration diagram for SERIAL_IO_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- SERIAL_IO_SPI_CONFIG SpiDeviceConfig [PCH_MAX_SERIALIO_SPI_CONTROLLERS]
  SPI Configuration.
- SERIAL_IO_I2C_CONFIG I2cDeviceConfig [PCH_MAX_SERIALIO_I2C_CONTROLLERS]
  I2C Configuration.
- SERIAL_IO_I3C_CONFIG I3cDeviceConfig [PCH_MAX_SERIALIO_I3C_CONTROLLERS]
  I3C Configuration.
- SERIAL_IO_UART_CONFIG UartDeviceConfig [PCH_MAX_SERIALIO_UART_CONTROLLERS]
  UART Configuration.

12.124.1 Detailed Description

The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers.

Revision 1:

- Inital version. **Revision 2:**
- Modified SPI Config to support pinmux functionality. **Revision 3:**
- Modified I3C Config to support more controllers.

Definition at line 61 of file SerialIoConfig.h.

The documentation for this struct was generated from the following file:

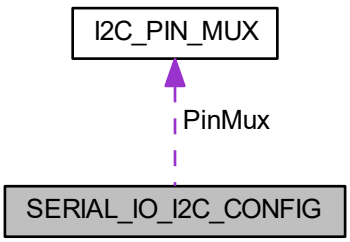
- SerialIoConfig.h

12.125 SERIAL_IO_I2C_CONFIG Struct Reference

Serial IO I2C Controller Configuration.

```
#include <SerialIoDevices.h>
```

Collaboration diagram for SERIAL_IO_I2C_CONFIG:



Public Attributes

- UINT8 PadTermination

  SerialIoI2cPci see SERIAL_IO_I2C_MODE
- I2C_PIN_MUX PinMux

  I2C pin mux configuration.

12.125.1 Detailed Description

Serial IO I2C Controller Configuration.

Definition at line 220 of file SerialIoDevices.h.

12.125.2 Member Data Documentation

12.125.2.1 PadTermination

`UINT8 SERIAL_IO_I2C_CONFIG::PadTermination`

SerialIoI2cPci see SERIAL_IO_I2C_MODE

I2C Pads Internal Termination. For more information please see Platform Design Guide. Supported values (check GPIO_ELECTRICAL_CONFIG for reference): **GpioTermNone: No termination**, GpioTermWpu1K: 1kOhm weak pull-up, GpioTermWpu5K: 5kOhm weak pull-up, GpioTermWpu20K: 20kOhm weak pull-up

Definition at line 231 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

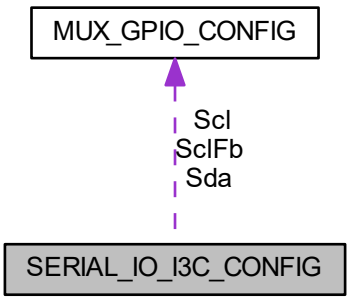
- SerialIoDevices.h

12.126 SERIAL_IO_I3C_CONFIG Struct Reference

The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller.

`#include <SerialIoDevices.h>`

Collaboration diagram for SERIAL_IO_I3C_CONFIG:



Public Attributes

- UINT8 Mode
  <b<SerialIoI3cPci ***see SERIAL_IO_I3C_MODE***
- UINT8 Reserved [3]
  Reserved bytes.

12.126.1 Detailed Description

The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller.

Definition at line 269 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

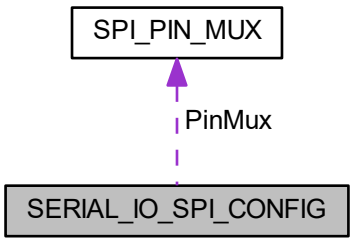
- SerialIoDevices.h

12.127 SERIAL_IO_SPI_CONFIG Struct Reference

The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.

`#include <SerialIoDevices.h>`

Collaboration diagram for SERIAL_IO_SPI_CONFIG:



Public Attributes

- UINT8 Mode
  SerialIoSpiPci *see SERIAL_IO_SPI_MODE*
- UINT8 DefaultCsOutput
  0 = CS0 *CS1, CS2, CS3. Default CS used by the SPI HC*
- UINT8 CsPolarity [PCH_MAX_SERIALIO_SPI_CHIP_SELECTS]
  Selects SPI ChipSelect signal polarity, 0 = low ***1 = High***
- UINT8 CsEnable [PCH_MAX_SERIALIO_SPI_CHIP_SELECTS]
  0 = Enable *1 = Disable. Based on this setting GPIO for given SPIx CSx will be configured in Native mode*
- UINT8 CsMode
  0 = HW Control *1 = SW Control. Sets Chip Select Control mode Hardware or Software.*
- UINT8 CsState
  0 = CS is set to low *1 = CS is set to high*
- SPI_PIN_MUX PinMux
  SPI Pinmux configuration.

12.127.1 Detailed Description

The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.

Definition at line 86 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

- SerialIoDevices.h

12.128 SERIAL_IO_UART_ATTRIBUTES Struct Reference

UART Settings.

```
#include <SerialIoDevices.h>
```

Public Attributes

- UINT32 BaudRate

  115200 *Max 6000000 MdePkg.dec PcdUartDefaultBaudRate*
- UINT8 Parity

  1 - No Parity *see EFI_PARITY_TYPE MdePkg.dec PcdUartDefaultParity*
- UINT8 DataBits

  8 *MdePkg.dec PcdUartDefaultDataBits*
- UINT8 StopBits

  1 - One Stop Bit *see EFI_STOP_BITS_TYPE MdePkg.dec PcdUartDefaultStopBits*
- UINT8 AutoFlow

  FALSE *IntelFrameworkModulePkg.dsc PcdIsaBusSerialUseHalfHandshake*

12.128.1 Detailed Description

UART Settings.

Definition at line 143 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

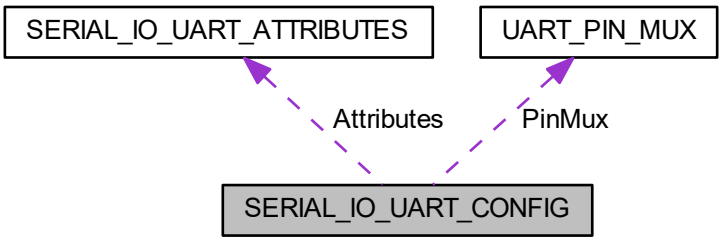
- SerialIoDevices.h

12.129 SERIAL_IO_UART_CONFIG Struct Reference

Serial IO UART Controller Configuration.

`#include <SerialIoDevices.h>`

Collaboration diagram for SERIAL_IO_UART_CONFIG:



Public Attributes

- SERIAL_IO_UART_ATTRIBUTES Attributes

  see SERIAL_IO_UART_ATTRIBUTES
- UART_PIN_MUX PinMux

  UART pin mux configuration.
- UINT8 Mode

  SerialIoUartPci *see SERIAL_IO_UART_MODE*
- UINT8 DBG2

  FALSE *If TRUE adds UART to DBG2 table and overrides UartPg to SerialIoUartPgDisabled*
- UINT8 PowerGating

  SerialIoUartPgAuto *Applies to Hidden/COM/SkipInit see SERIAL_IO_UART_PG*
- UINT8 DmaEnable

  TRUE *Applies to SerialIoUartPci only. Informs OS driver to use DMA, if false it will run in PIO mode*

12.129.1 Detailed Description

Serial IO UART Controller Configuration.

Definition at line 166 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

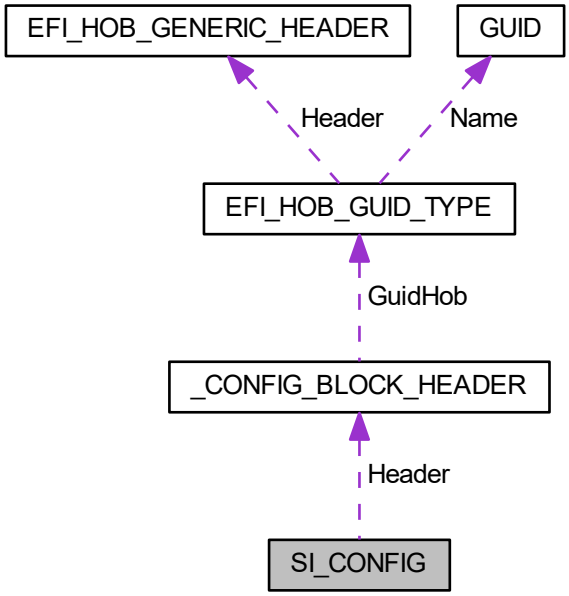
- SerialIoDevices.h

12.130 SI_CONFIG Struct Reference

The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

`#include <SiConfig.h>`

Collaboration diagram for SI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0 - 27 Config Block Header.
- UINT8 SkipSsidProgramming

  This is used to skip the SSID programming in silicon code.
- UINT8 RsvdBytes0 [3]

  offset 29 - 31
- UINT16 CustomizedSvid

  When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the SVID for all internal devices.
- UINT16 CustomizedSsid

  When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the Sid for all internal devices.
- UINT32 * SsidTablePtr

  SsidTablePtr contains the SVID_SID_INIT_ENTRY table.
- UINT16 NumberOfSsidTableEntry

  Number of valid enties in SsidTablePtr.

- UINT8 RsvdBytes1 [2]

  offset 42 - 43
- UINT8 SkipBiosDoneWhenFwUpdate

  This is used to skip setting BIOS_DONE MSR during firmware update boot mode.
- UINT8 RsvdBytes2 [3]

  Offset 45 - 47.

12.130.1 Detailed Description

The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

Revision 1: - Initial version.

Definition at line 51 of file SiConfig.h.

12.130.2 Member Data Documentation

12.130.2.1 CustomizedSsid

```
UINT16 SI_CONFIG::CustomizedSsid
```

When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the Sid for all internal devices.

0: use silicon default SSID 0x7270 , Non-zero: use customized SSID. offset 34 - 35

Definition at line 75 of file SiConfig.h.

12.130.2.2 CustomizedSvid

```
UINT16 SI_CONFIG::CustomizedSvid
```

When SkipSsidProgramming is FALSE, silicon code will use this as default value to program the SVID for all internal devices.

0: use silicon default SVID 0x8086 , Non-zero: use customized SVID. offset 32 - 33

Definition at line 69 of file SiConfig.h.

12.130.2.3 NumberOfSsidTableEntry

`UINT16 SI_CONFIG::NumberOfSsidTableEntry`

Number of valid enties in SsidTablePtr.

This is valid when SkipSsidProgramming is FALSE; **Default is 0.** offset 40 - 41

Definition at line 114 of file SiConfig.h.

12.130.2.4 SkipBiosDoneWhenFwUpdate

`UINT8 SI_CONFIG::SkipBiosDoneWhenFwUpdate`

This is used to skip setting BIOS_DONE MSR during firmware update boot mode.

When set to TRUE and boot mode is BOOT_ON_FLASH_UPDATE, skip setting BIOS_DONE MSR at EndofPei. **0: FALSE**, 1: TRUE Offset 44

Definition at line 122 of file SiConfig.h.

12.130.2.5 SkipSsidProgramming

`UINT8 SI_CONFIG::SkipSsidProgramming`

This is used to skip the SSID programming in silicon code.

When set to TRUE, silicon code will not do any SSID programming and platform code needs to handle that by itself properly. **0: FALSE**, 1: TRUE offset 28

Definition at line 62 of file SiConfig.h.

12.130.2.6 SsidTablePtr

`UINT32* SI_CONFIG::SsidTablePtr`

SsidTablePtr contains the SVID_SID_INIT_ENTRY table.

This is valid when SkipSsidProgramming is FALSE; It doesn't need to contain entries for all Intel internal devices. It can only contains the SVID_SID_INIT_ENTRY entries for those Dev# Func# which needs to be overridden. In the enties, only Dev, Function, SubSystemVendorId, and SubSystemId are required. **Default is NULL.**

E.g. Platform only needs to override BDF 0:31:5 to AAAA:BBBB and BDF 0:31:3 to CCCC:DDDD, it can be done in platform like this: STATIC SVID_SID_INIT_ENTRY mSsidTablePtr[SI_MAX_DEVICE_COUNT] = {0};

VOID SiPolicyUpdate () { UINT32 EntryCount = 0; SiPolicy->SkipSsidProgramming = FALSE; SiPolicy->Ssid↩ TablePtr = mSsidTablePtr;

mSsidTablePtr[EntryCount].Address.Bits.Device = SpiDeviceNumber (); mSsidTablePtr[EntryCount].Address.↩ Bits.Function = SpiFunctionNumber (); mSsidTablePtr[EntryCount].SvidSidValue.SubSystemVendorId = 0xAA↩ AA; mSsidTablePtr[EntryCount].SvidSidValue.SubSystemId = 0xBBBB; EntryCount ++; mSsidTablePtr[Entry↩ Count].Address.Bits.Device = HdaDevNumber (); mSsidTablePtr[EntryCount].Address.Bits.Function = HdaFunc↩ Number (); mSsidTablePtr[EntryCount].SvidSidValue.SubSystemVendorId = 0xCCCC; mSsidTablePtr[Entry↩ Count].SvidSidValue.SubSystemId = 0xDDDD; EntryCount ++; ASSERT (EntryCount < SI_MAX_DEVICE_CO↩ UNT); SiPolicy->NumberOfSsidTableEntry = EntryCount; } offset 36 - 39

Definition at line 108 of file SiConfig.h.

The documentation for this struct was generated from the following file:

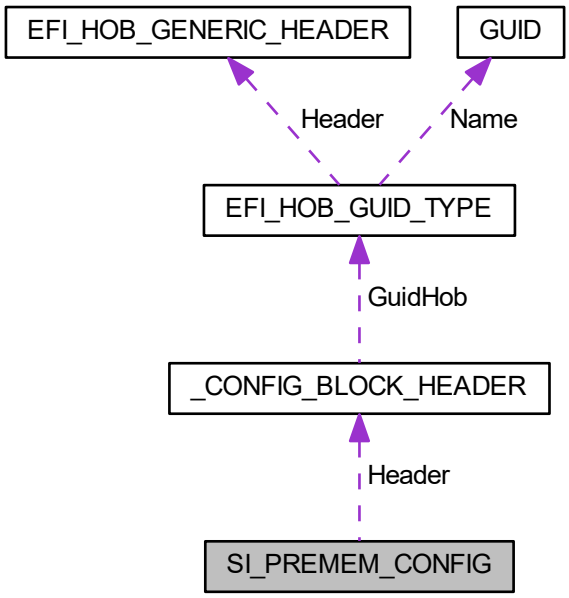
- SiConfig.h

12.131 SI_PREMEM_CONFIG Struct Reference

The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

`#include <SiPreMemConfig.h>`

Collaboration diagram for SI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0 - 27 Config Block Header.
- UINT32 PlatformDebugOption: 4

  Platform Debug Option As a main switch to enable platform debug capability and relevant settings.
- UINT32 RsvdBits: 28

  offset 28 - 31
- UINT8 SkipOverrideBootModeWhenFwUpdate

  This is used to skip override boot mode during firmware update boot mode.
- UINT8 RsvdBytes [3]

  offset 32

12.131.1 Detailed Description

The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

Revision 1:

- Initial version.

Definition at line 58 of file SiPreMemConfig.h.

12.131.2 Member Data Documentation

12.131.2.1 PlatformDebugOption

`UINT32 SI_PREMEM_CONFIG::PlatformDebugOption`

Platform Debug Option As a main switch to enable platform debug capability and relevant settings.

Manual: Do not use Platform Debug Option to override other debug-relevant policies, but the user must set each debug option manually, aimed at advanced users.

PlatformDebugOption related policies please refer to PLATFORM_DEBUG_OPTION **0:Disabled**; 2:Enabled TraceHub active; 4:Enabled TraceHub ready; 6:Enabled TraceHub power off; 7:Manual

Definition at line 68 of file SiPreMemConfig.h.

12.131.2.2 SkipOverrideBootModeWhenFwUpdate

`UINT8 SI_PREMEM_CONFIG::SkipOverrideBootModeWhenFwUpdate`

This is used to skip override boot mode during firmware update boot mode.

When set to TRUE and boot mode is BOOT_ON_FLASH_UPDATE, skip setting boot mode to BOOT_WITH_FU↩LL_CONFIGURATION in PEI memory init. **0: FALSE**, 1: TRUE

Definition at line 76 of file SiPreMemConfig.h.

The documentation for this struct was generated from the following file:

- SiPreMemConfig.h

12.132 SPI_PIN_MUX Struct Reference

SPI signals pin muxing settings.

`#include <SerialIoDevices.h>`

Public Attributes

- UINT32 Cs [PCH_MAX_SERIALIO_SPI_CHIP_SELECTS]
  CS Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_CS_∗.
- UINT32 Clk
  CLK Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_CLK_∗.
- UINT32 Miso
  MISO Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_MISO_∗.
- UINT32 Mosi
  MOSI Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_SPIx_MOSI_∗.

12.132.1 Detailed Description

SPI signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_SPIx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 76 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

- SerialIoDevices.h

12.133 SVID_SID_VALUE Struct Reference

Subsystem Vendor ID / Subsystem ID.

`#include <SiConfig.h>`

12.133.1 Detailed Description

Subsystem Vendor ID / Subsystem ID.

Definition at line 135 of file SiConfig.h.

The documentation for this struct was generated from the following file:

- SiConfig.h

12.134 TCSS_DEVEN_PEI_PREMEM_CONFIG Union Reference

The TCSS_DEVEN_PEI_PREMEM_CONFIG block describes Device Enable settings for TCSS.

`#include <TcssPeiPreMemConfig.h>`

Public Attributes

- UINT32 TcssDevEn
  Maps to bits in TCSS_DEVEN_0_0_0_MCHBAR_IMPH.

12.134.1 Detailed Description

The TCSS_DEVEN_PEI_PREMEM_CONFIG block describes Device Enable settings for TCSS.

Definition at line 48 of file TcssPeiPreMemConfig.h.

The documentation for this union was generated from the following file:

- TcssPeiPreMemConfig.h

12.135 TCSS_IOM_ORI_OVERRIDE Struct Reference

The TCSS_IOM_PEI_CONFIG block describes IOM Aux/HSL override settings for TCSS.

`#include <TcssPeiConfig.h>`

Public Attributes

- UINT16 AuxOri

  Bits defining value for IOM Aux Orientation Register.
- UINT16 HslOri

  Bits defining value for IOM HSL Orientation Register.

12.135.1 Detailed Description

The TCSS_IOM_PEI_CONFIG block describes IOM Aux/HSL override settings for TCSS.

Definition at line 156 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

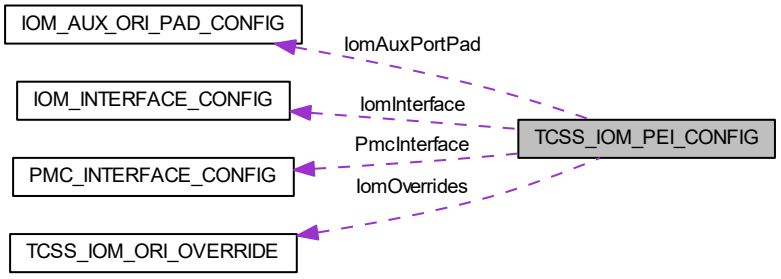
- TcssPeiConfig.h

12.136 TCSS_IOM_PEI_CONFIG Struct Reference

The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS.

`#include <TcssPeiConfig.h>`

Collaboration diagram for TCSS_IOM_PEI_CONFIG:



Public Attributes

- IOM_AUX_ORI_PAD_CONFIG IomAuxPortPad [MAX_IOM_AUX_BIAS_COUNT]

  The IOM_AUX_ORI_BIAS_CTRL port config setting.
- IOM_INTERFACE_CONFIG IomInterface

  Config settings are BIOS <-> IOM interface.
- PMC_INTERFACE_CONFIG PmcInterface

  Config settings for BIOS <-> PMC interface.
- UINT8 TcStateLimit

  Tcss C-State deep stage.
- UINT8 TcNotifyIgd

  Wake iGfx from D3 on HPD_ASSERT or HPD_IRQ assocaited with the DP_ALT and DP tunneling connection.
- UINT8 Reserved [2]

  Reserved bytes for future use.

12.136.1 Detailed Description

The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS.

Definition at line 164 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

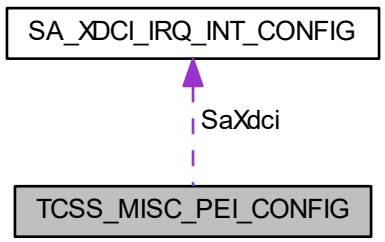
- TcssPeiConfig.h

12.137 TCSS_MISC_PEI_CONFIG Struct Reference

The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS.

```
#include <TcssPeiConfig.h>
```

Collaboration diagram for TCSS_MISC_PEI_CONFIG:



Public Attributes

- SA_XDCI_IRQ_INT_CONFIG SaXdci

  System Agent Xdci Int Pin and Irq setting.
- TCSS_TYPEC_CONV_TYPEA TcssConvTypeA [MAX_TCSS_USB3_PORTS]

  TCSS Port Convert to TypeA.
- UINT8 Reserved [8]

  Reserved bytes for future use.

12.137.1 Detailed Description

The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS.

Definition at line 177 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

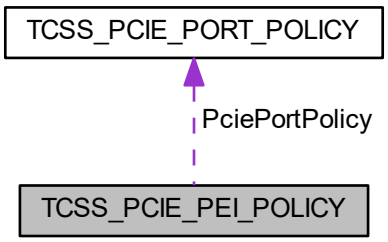
- TcssPeiConfig.h

12.138 TCSS_PCIE_PEI_POLICY Struct Reference

TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS.

```
#include <TcssPeiConfig.h>
```

Collaboration diagram for TCSS_PCIE_PEI_POLICY:



Public Attributes

- UINT8 Reserved [4]

  Reserved bytes for future use.

12.138.1 Detailed Description

TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS.

Definition at line 148 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiConfig.h

12.139 TCSS_PCIE_PORT_POLICY Struct Reference

The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS.

```
#include <TcssPeiConfig.h>
```

Public Attributes

- UINT8 AcsEnabled

  *Indicate whether the ACS is enabled. 0: Disable; **1: Enable**.*
- UINT8 LtrEnable

  *Latency Tolerance Reporting Mechanism. **0: Disable**; 1: Enable.*
- UINT8 PtmEnabled

  Enables PTM capability.
- UINT8 Aspm

  The ASPM configuration of the root port (see: PCH_PCIE_ASPM_CONTROL). Default is
- UINT8 SlotNumber

  Indicates the slot number for the root port. Default is the value as root port index.
- UINT8 SlotPowerLimitScale

  ***(Test)** Specifies scale used for slot power limit value. Leave as 0 to set to default. Default is **zero**.*
- UINT16 SlotPowerLimitValue

  ***(Test)** Specifies upper limit on power supplies by slot. Leave as 0 to set to default. Default is **zero**.*
- UINT8 AdvancedErrorReporting

  *Indicate whether the Advanced Error Reporting is enabled. **0: Disable**; 1: Enable.*
- UINT8 UnsupportedRequestReport

  *Indicate whether the Unsupported Request Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 FatalErrorReport

  *Indicate whether the Fatal Error Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 NoFatalErrorReport

  *Indicate whether the No Fatal Error Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 CorrectableErrorReport

  *Indicate whether the Correctable Error Report is enabled. **0: Disable**; 1: Enable.*
- UINT8 SystemErrorOnFatalError

  *Indicate whether the System Error on Fatal Error is enabled. **0: Disable**; 1: Enable.*
- UINT8 SystemErrorOnNonFatalError

  *Indicate whether the System Error on Non Fatal Error is enabled. **0: Disable**; 1: Enable.*
- UINT8 SystemErrorOnCorrectableError

  *Indicate whether the System Error on Correctable Error is enabled. **0: Disable**; 1: Enable.*
- UINT16 LtrMaxSnoopLatency

Latency Tolerance Reporting, Max Snoop Latency.

- UINT16 LtrMaxNoSnoopLatency

  Latency Tolerance Reporting, Max Non-Snoop Latency.
- UINT8 SnoopLatencyOverrideMode

  Latency Tolerance Reporting, Snoop Latency Override Mode.
- UINT8 SnoopLatencyOverrideMultiplier

  Latency Tolerance Reporting, Snoop Latency Override Multiplier.
- UINT16 SnoopLatencyOverrideValue

  Latency Tolerance Reporting, Snoop Latency Override Value.
- UINT8 NonSnoopLatencyOverrideMode

  Latency Tolerance Reporting, Non-Snoop Latency Override Mode.
- UINT8 NonSnoopLatencyOverrideMultiplier

  Latency Tolerance Reporting, Non-Snoop Latency Override Multiplier.
- UINT16 NonSnoopLatencyOverrideValue

  Latency Tolerance Reporting, Non-Snoop Latency Override Value.
- UINT8 ForceLtrOverride

  ***0: Disable**; 1: Enable.*
- UINT8 LtrConfigLock

  ***0: Disable**; 1: Enable.*

12.139.1 Detailed Description

The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS.

Definition at line 113 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

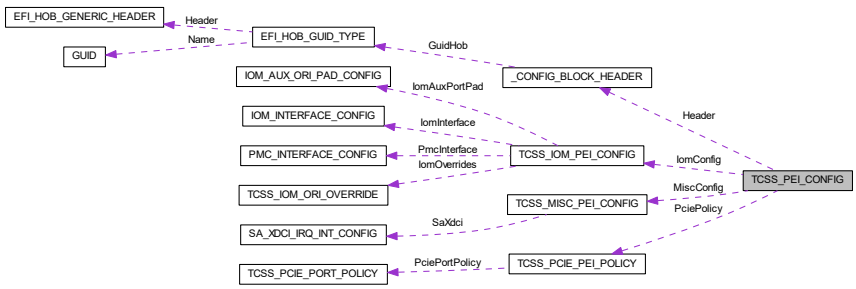
- TcssPeiConfig.h

12.140 TCSS_PEI_CONFIG Struct Reference

The TCSS_PEI_CONFIG block describes TCSS settings for SA.

`#include <TcssPeiConfig.h>`

Collaboration diagram for TCSS_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- TCSS_PCIE_PEI_POLICY PciePolicy

  The PCIe Config.
- USB_CONFIG UsbConfig

  USB config is shared between PCH and SA.
- TCSS_IOM_PEI_CONFIG IomConfig

  The Iom Config.
- TCSS_MISC_PEI_CONFIG MiscConfig

  The MISC Config.

12.140.1 Detailed Description

The TCSS_PEI_CONFIG block describes TCSS settings for SA.

Definition at line 186 of file TcssPeiConfig.h.

The documentation for this struct was generated from the following file:

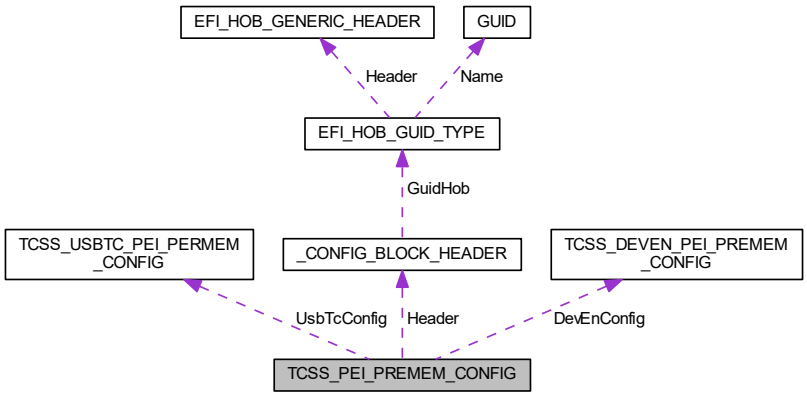
- TcssPeiConfig.h

12.141 TCSS_PEI_PREMEM_CONFIG Struct Reference

This configuration block describes TCSS settings.

```
#include <TcssPeiPreMemConfig.h>
```

Collaboration diagram for TCSS_PEI_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- TCSS_DEVEN_PEI_PREMEM_CONFIG DevEnConfig

  TCSS DEVEN.
- TCSS_USBTC_PEI_PERMEM_CONFIG UsbTcConfig

  USB Type C Port Configuration.

12.141.1 Detailed Description

This configuration block describes TCSS settings.

Revision 1:

- Initial version.

Definition at line 77 of file TcssPeiPreMemConfig.h.

The documentation for this struct was generated from the following file:

- TcssPeiPreMemConfig.h

12.142 TCSS_USBTC_PEI_PERMEM_CONFIG Struct Reference

The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.

```
#include <TcssPeiPreMemConfig.h>
```

Public Attributes

- UINT32 UsbTcPortEn: 4

  bitmap for USB Type C enabled ports
- UINT32 Rsvd: 28

  Reserved bytes for future use.

12.142.1 Detailed Description

The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.

Definition at line 67 of file TcssPeiPreMemConfig.h.

The documentation for this struct was generated from the following file:

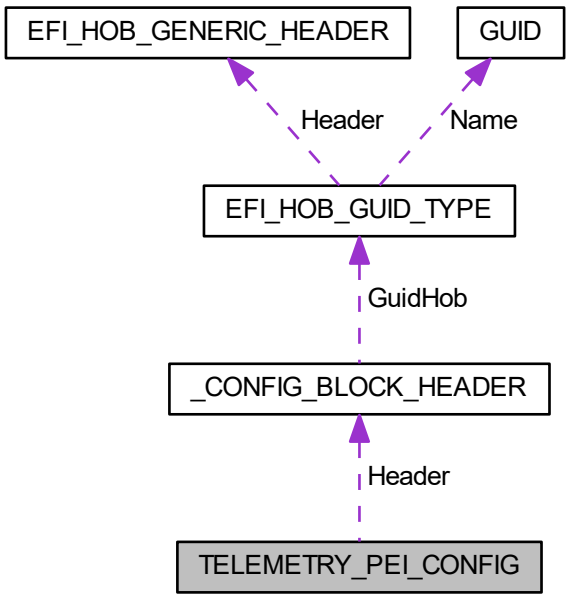
- TcssPeiPreMemConfig.h

12.143 TELEMETRY_PEI_CONFIG Struct Reference

This configuration block describes Telemetry settings in PostMem.

`#include <TelemetryPeiConfig.h>`

Collaboration diagram for TELEMETRY_PEI_CONFIG:



Public Attributes

- UINT32 CpuCrashLogEnable
  Config Block Header.

12.143.1 Detailed Description

This configuration block describes Telemetry settings in PostMem.

Revision 1:

- Initial version.

Definition at line 63 of file TelemetryPeiConfig.h.

The documentation for this struct was generated from the following file:

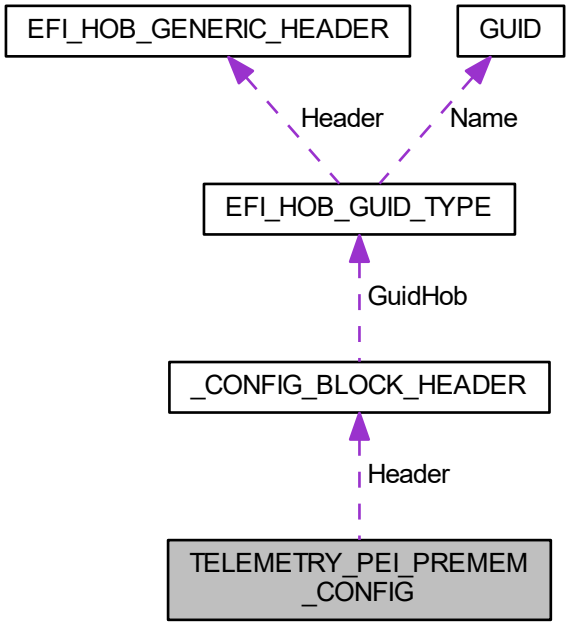
- TelemetryPeiConfig.h

12.144 TELEMETRY_PEI_PREMEM_CONFIG Struct Reference

This configuration block describes Telemetry settings in PreMem.

`#include <TelemetryPeiConfig.h>`

Collaboration diagram for TELEMETRY_PEI_PREMEM_CONFIG:



Public Attributes

- UINT32 CpuCrashLogDevice
  Config Block Header.

12.144.1 Detailed Description

This configuration block describes Telemetry settings in PreMem.

Revision 1:

- Initial version.

Definition at line 53 of file TelemetryPeiConfig.h.

The documentation for this struct was generated from the following file:

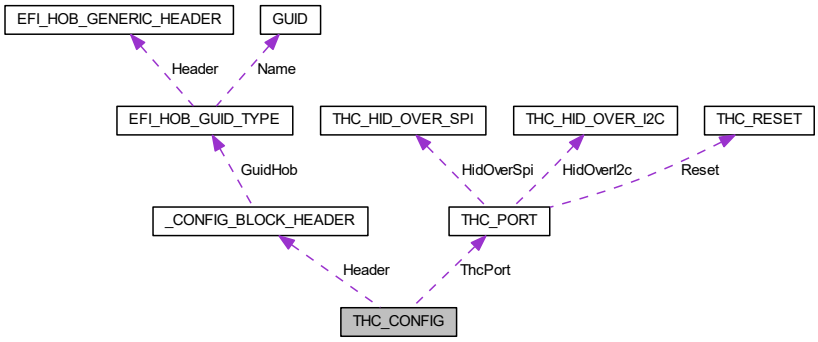
- TelemetryPeiConfig.h

12.145 THC_CONFIG Struct Reference

THC_CONFIG block provides the configurations forTouch Host Controllers.

`#include <ThcConfig.h>`

Collaboration diagram for THC_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- THC_PORT ThcPort [2]
  Port Configuration.

12.145.1 Detailed Description

THC_CONFIG block provides the configurations forTouch Host Controllers.

Assignment field in each THC port controlls the THC behavior.

Available scenarios: 1: Single Port 0 used by THC0

- THC0 Enabled
- Port0 assigned to THC0
- Port1 unassigned
- THC1 will be automatically Disabled. 2: Both ports used by THC0
- THC0 Enabled
- Port0 assigned to THC0
- Port1 assigned to THC0
- THC1 will be automatically Disabled. 3: Port 0 used by THC0 and Port 1 used by THC1
- THC0 Enabled
- Port0 assigned to THC0
- THC1 Enabled
- Port1 assigned to THC1. **4: Both Ports unassigned.** Both THC Controllers will be disabled in that case.

Note

Invalid scenario that will cause ASSERT.

1. Same port Number assigned to THC0 or THC1.
2. Two Ports assigned to THC1.

Revision 1:

- Inital version. **Revision 2:**
- Add HidOverSpi THC_MODE **Revision 3:**
- Expand HidOverSpi support **Revision 4:**
- Add ResetSequencingDelay in THC_HID_OVER_SPI **Revision 5:**
- Moved out reset data from THC_HID_OVER_SPI
- Add THC_HID_OVER_I2C

Definition at line 196 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.146 THC_HID_OVER_I2C Struct Reference

THC Hid Over I2C mode related settings.

`#include <ThcConfig.h>`

Public Attributes

- UINT32 DeviceAddress

  Device Address.
- UINT32 ConnectionSpeed

  Connection Speed - Frequency [Hz].
- UINT32 AddressingMode

  Addressing Mode 0x1: The connection uses 10-bit addressing.
- UINT32 StandardModeSerialClockLineHighPeriod

  Serial Clock Line High Period.
- UINT32 StandardModeSerialClockLineLowPeriod

  Standard Mode Serial Clock Line Low Period.
- UINT32 StandardModeSerialDataLineTransmitHoldPeriod

  Standard Mode Serial Data Line Transmit Hold Period.
- UINT32 StandardModeSerialDataLineReceiveHoldPeriod

  Standard Mode Serial Data Line Receive Hold Period.
- UINT32 FastModeSerialClockLineHighPeriod

  Fast Mode Serial Clock Line High Period.
- UINT32 FastModeSerialClockLineLowPeriod

Fast Mode Serial Clock Line Low Period.

- UINT32 FastModeSerialDataLineTransmitHoldPeriod

  Fast Mode Serial Data Line Transmit Hold Period.
- UINT32 FastModeSerialDataLineReceiveHoldPeriod

  Fast Mode Serial Data Line Receive Hold Period.
- UINT32 MaxSuppressedSpikesSMFMFMP

  Maximum Length Of Suppressed Spikes In Std Mode Fast Mode And Fast Mode Plus.
- UINT32 FastModePlusSerialClockLineHighPeriod

  Fast Mode Plus Serial Clock Line High Period.
- UINT32 FastModePlusSerialClockLineLowPeriod

  Fast Mode Plus Serial Clock Line Low Period.
- UINT32 FastModePlusSerialDataLineTransmitHoldPeriod

  Fast Mode Plus Serial Data Line Transmit Hold Period.
- UINT32 FastModePlusSerialDataLineReceiveHoldPeriod

  Fast Mode Plus Serial Data Line Receive Hold Period.
- UINT32 HighSpeedModePlusSerialClockLineHighPeriod

  High Speed Mode Plus Serial Clock Line High Period.
- UINT32 HighSpeedModePlusSerialClockLineLowPeriod

  High Speed Mode Plus Serial Clock Line Low Period.
- UINT32 HighSpeedModePlusSerialDataLineTransmitHoldPeriod

  High Speed Mode Plus Serial Data Line Transmit Hold Period.
- UINT32 HighSpeedModePlusSerialDataLineReceiveHoldPeriod

  High Speed Mode Plus Serial Data Line Receive Hold Period.
- UINT32 MaximumLengthOfSuppressedSpikesInHighSpeedMode

  Maximum Length Of Suppressed Spikes In High Speed Mode.

12.146.1 Detailed Description

THC Hid Over I2C mode related settings.

Only applicable when HidOverI2c is enbled.

Definition at line 96 of file ThcConfig.h.

12.146.2 Member Data Documentation

12.146.2.1 AddressingMode

```
UINT32 THC_HID_OVER_I2C::AddressingMode
```

Addressing Mode 0x1: The connection uses 10-bit addressing.

0x0: The connection uses 7-bit addressing.

Definition at line 104 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.147 THC_HID_OVER_SPI Struct Reference

THC Hid Over SPI mode related settings.

`#include <ThcConfig.h>`

Public Attributes

- UINT32 Frequency
  Connection Speed - SPI Frequency in Hz.
- UINT32 InputReportHeaderAddress
  Input Report Header Address.
- UINT32 InputReportBodyAddress
  Input Report Body Address.
- UINT32 OutputReportAddress
  Output Report Address.
- UINT32 ReadOpcode
  Read Opcode.
- UINT32 WriteOpcode
  Write Opcode.
- UINT32 Flags
  Flags.
- UINT32 LimitPacketSize
  Limit Packet Size.

12.147.1 Detailed Description

THC Hid Over SPI mode related settings.

Only applicable when HidOverSpi is enbled.

Definition at line 73 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

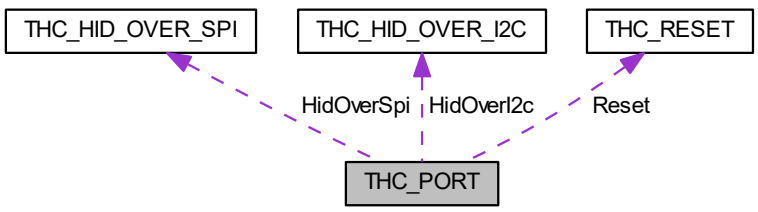
- ThcConfig.h

12.148 THC_PORT Struct Reference

Port Configuration structure required for each Port that THC might use.

`#include <ThcConfig.h>`

Collaboration diagram for THC_PORT:



Public Attributes

- UINT32 Assignment

  Sets THCx assignment see THC_PORT_ASSIGNMENT.
- UINT32 InterruptPinMuxing

  Each GPIO PORTx/SPIx INTB Pin has different muxing options refer to GPIO_*_MUXING_THC_SPIx_*.
- UINT8 WakeOnTouch

  *1 = Enable **0 = Disable**. Based on this setting vGPIO for given THC will be in native mode, and additional _CRS for wake will be exposed in ACPI*
- UINT8 Mode

  *1 = Hid over SPI **0 = THC**. Switch between Intel THC protcol and Industry standard HID Over SPI protocol.*
- UINT8 TimestampTimerMode

  *Timestamp timer behavior in D0i2 **1 = Timer resets to 0 when entering D0i2** 0 = Timer is paused instead of reset to 0 when entering D0i2.*
- UINT8 Reserved [1]

  Reserved.
- THC_HID_OVER_SPI HidOverSpi

  Hid Over Spi mode settings.
- UINT32 ActiveLtr

  ***0xFFFFFFFF = Driver Default** Active Ltr*
- UINT32 IdleLtr

  ***0xFFFFFFFF = Driver Default** Idle Ltr*
- UINT32 PerformanceLimitation

  Performance Limitation.
- UINT32 DisplayFrameSyncPeriod

  *Period of the emulated display frame sync [ms] The minimum period is 2ms, maximum period is 100ms Default: **10**ms.*
- THC_RESET Reset

  Reset Line settings.
- THC_HID_OVER_I2C HidOverI2c

  Hid Over I2c mode settings.

12.148.1 Detailed Description

Port Configuration structure required for each Port that THC might use.

Definition at line 128 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

- ThcConfig.h

12.149 THC_RESET Struct Reference

THC Reset Pad related settings.

```
#include <ThcConfig.h>
```

Public Attributes

- UINT32 ResetPad

  *Reset Pad - **0x0 = THC HW default** BIOS will pass Pad infromation to ACPI, this Pad is used in _RST function.*
- UINT32 ResetPadTrigger

  Reset Pad Trigger.
- UINT32 ResetSequencingDelay

  Policy control for reset sequencing delay (_INI, _RST)

12.149.1 Detailed Description

THC Reset Pad related settings.

Definition at line 87 of file ThcConfig.h.

The documentation for this struct was generated from the following file:

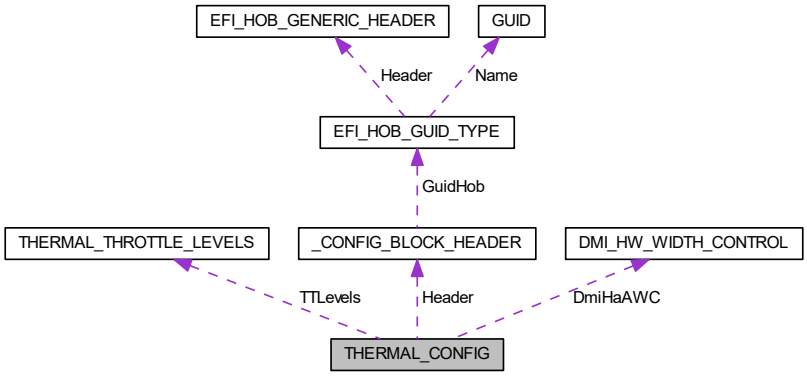
- ThcConfig.h

12.150 THERMAL_CONFIG Struct Reference

The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block.

`#include <ThermalConfig.h>`

Collaboration diagram for THERMAL_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- UINT32 PchHotEnable: 1

  *Enable PCHHOT# pin assertion when temperature is higher than PchHotLevel. 0: **Disabled, 1: Enabled.***
- THERMAL_THROTTLE_LEVELS TTLevels

  This field decides the settings of Thermal throttling.
- DMI_HW_WIDTH_CONTROL DmiHaAWC

  This field decides the settings of DMI throttling.
- UINT16 PchHotLevel

  The recommendation is the same as Cat Trip point.

12.150.1 Detailed Description

The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block.

Definition at line 97 of file ThermalConfig.h.

12.150.2 Member Data Documentation

12.150.2.1 DmiHaAWC

```
DMI_HW_WIDTH_CONTROL THERMAL_CONFIG::DmiHaAWC
```

This field decides the settings of DMI throttling.

When the Suggested Setting is enabled, PCH RC will use the suggested representative values.

Definition at line 110 of file ThermalConfig.h.

12.150.2.2 PchHotLevel

```
UINT16 THERMAL_CONFIG::PchHotLevel
```

The recommendation is the same as Cat Trip point.

This field decides the temperature, default is **120**. Temperature value used for PCHHOT# pin assertion based on 2s complement format

- 0x001 positive 1'C
- 0x000 0'C
- 0x1FF negative 1'C
- 0x1D8 negative 40'C
- and so on

Definition at line 121 of file ThermalConfig.h.

12.150.2.3 TTLevels

```
THERMAL_THROTTLE_LEVELS THERMAL_CONFIG::TTLevels
```

This field decides the settings of Thermal throttling.

When the Suggested Setting is enabled, PCH RC will use the suggested representative values.

Definition at line 105 of file ThermalConfig.h.

The documentation for this struct was generated from the following file:

- ThermalConfig.h

12.151 THERMAL_THROTTLE_LEVELS Struct Reference

This structure lists PCH supported throttling register setting for custimization.

```
#include <ThermalConfig.h>
```

Public Attributes

- UINT32 T0Level: 9

  Custimized T0Level value. If SuggestedSetting is used, this setting is ignored.
- UINT32 T1Level: 9

  Custimized T1Level value. If SuggestedSetting is used, this setting is ignored.
- UINT32 T2Level: 9

  Custimized T2Level value. If SuggestedSetting is used, this setting is ignored.
- UINT32 TTEnable: 1

  Enable the thermal throttle function. If SuggestedSetting is used, this settings is ignored.
- UINT32 TTState13Enable: 1

  When set to 1 and the programmed GPIO pin is a 1, then PMSync state 13 will force at least T2 state.
- UINT32 TTLock: 1

  When set to 1, this entire register (TL) is locked and remains locked until the next platform reset.
- UINT32 SuggestedSetting: 1

  0: Disable; ***1: Enable*** *suggested representative values.*
- UINT32 RsvdBits0: 1

  Reserved bits.
- UINT32 Rsvd0

  Reserved bytes.

12.151.1 Detailed Description

This structure lists PCH supported throttling register setting for custimization.

When the SuggestedSetting is enabled, the customized values are ignored.

Definition at line 47 of file ThermalConfig.h.

12.151.2 Member Data Documentation

12.151.2.1 TTLock

`UINT32 THERMAL_THROTTLE_LEVELS::TTLock`

When set to 1, this entire register (TL) is locked and remains locked until the next platform reset.

If SuggestedSetting is used, this setting is ignored.

Definition at line 61 of file ThermalConfig.h.

12.151.2.2 TTState13Enable

`UINT32 THERMAL_THROTTLE_LEVELS::TTState13Enable`

When set to 1 and the programmed GPIO pin is a 1, then PMSync state 13 will force at least T2 state.

If SuggestedSetting is used, this setting is ignored.

Definition at line 56 of file ThermalConfig.h.

The documentation for this struct was generated from the following file:

- ThermalConfig.h

12.152 TRACE_HUB_CONFIG Struct Reference

TRACE_HUB_CONFIG block describes TraceHub settings.

`#include <TraceHubConfig.h>`

Public Attributes

- UINT64 MemReg0Size
  Trace hub memory buffer region size value.
- UINT8 EnableMode
  Trace hub mode.
- UINT8 AetEnabled
  AET is one of FW trace agents which is exclusively rounted to one TraceHub.
- UINT8 BiosTraceSink
  *Indicate if BIOS trace is sent to this TraceHub instance **0 = Disable**; 1 = Enabled.*
- UINT8 Reserved [5]
  Filling reserved for future use & Config block DW alignment.

12.152.1 Detailed Description

TRACE_HUB_CONFIG block describes TraceHub settings.

Definition at line 84 of file TraceHubConfig.h.

12.152.2 Member Data Documentation

12.152.2.1 AetEnabled

```
UINT8 TRACE_HUB_CONFIG::AetEnabled
```

AET is one of FW trace agents which is exclusively rounted to one TraceHub.

i.e. when one TraceHub device enableds it, the other one must disable it. **0 = Disable**; 1 = Enabled

Definition at line 103 of file TraceHubConfig.h.

12.152.2.2 EnableMode

```
UINT8 TRACE_HUB_CONFIG::EnableMode
```

Trace hub mode.

Default is disabled. **0 = Disable**; 1 = Enabled Refer to TRACE_HUB_ENABLE_MODE

Definition at line 97 of file TraceHubConfig.h.

12.152.2.3 MemReg0Size

```
UINT64 TRACE_HUB_CONFIG::MemReg0Size
```

Trace hub memory buffer region size value.

Supported size refer to TRACE_BUFFER_SIZE

Definition at line 90 of file TraceHubConfig.h.

The documentation for this struct was generated from the following file:

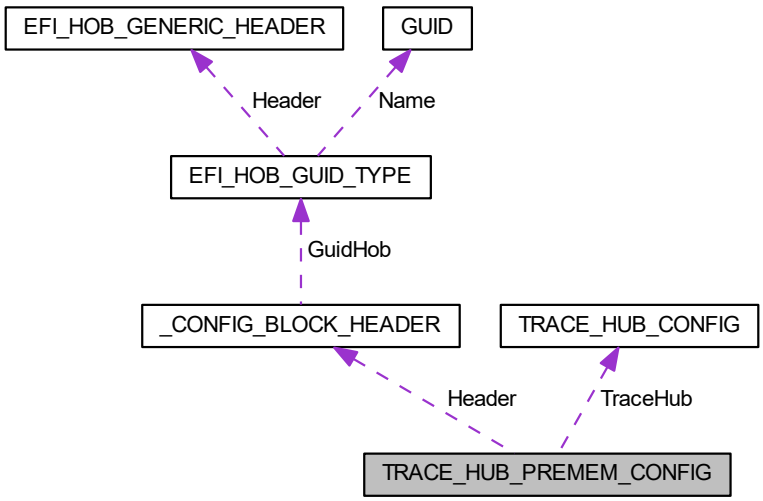
- TraceHubConfig.h

12.153 TRACE_HUB_PREMEM_CONFIG Struct Reference

Trace Hub PreMem Configuration Contains Trace Hub settings **Revision 1**: - Initial version.

`#include <TraceHubConfig.h>`

Collaboration diagram for TRACE_HUB_PREMEM_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header
  Config Block Header.
- TRACE_HUB_CONFIG TraceHub [MaxTraceHub]
  Trace Hub Config.

12.153.1 Detailed Description

Trace Hub PreMem Configuration Contains Trace Hub settings **Revision 1**: - Initial version.

Definition at line 131 of file TraceHubConfig.h.

The documentation for this struct was generated from the following file:

- TraceHubConfig.h

12.154 UART_PIN_MUX Struct Reference

UART signals pin muxing settings.

`#include <SerialIoDevices.h>`

Public Attributes

- UINT32 Rx

  RXD Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_RXD_∗.
- UINT32 Tx

  TXD Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_TXD_∗.
- UINT32 Rts

  RTS Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_RTS_∗.
- UINT32 Cts

  CTS Pin mux configuration. Refer to GPIO_∗_MUXING_SERIALIO_UARTx_CTS_∗.

12.154.1 Detailed Description

UART signals pin muxing settings.

If signal can be enable only on a single pin then this parameter is ignored by RC. Refer to GPIO_∗_MUXING_SE↩ RIALIO_UARTx_∗ in GpioPins∗.h for supported settings on a given platform

Definition at line 156 of file SerialIoDevices.h.

The documentation for this struct was generated from the following file:

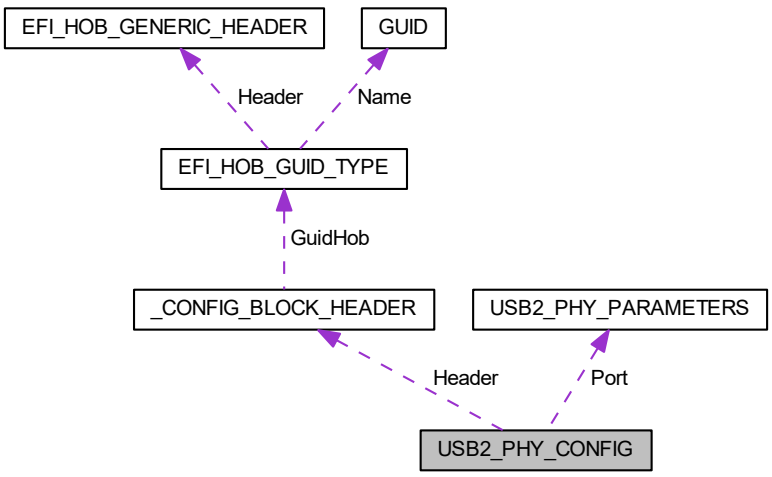
- SerialIoDevices.h

12.155 USB2_PHY_CONFIG Struct Reference

This structure holds info on how to tune electrical parameters of USB2 ports based on board layout.

`#include <Usb2PhyConfig.h>`

Collaboration diagram for USB2_PHY_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Config Block Header.
- USB2_PHY_PARAMETERS Port [MAX_USB2_PORTS]

  This structure configures per USB2 port physical settings.

12.155.1 Detailed Description

This structure holds info on how to tune electrical parameters of USB2 ports based on board layout.

Revision 1:

- Initial version.

Definition at line 102 of file Usb2PhyConfig.h.

12.155.2 Member Data Documentation

12.155.2.1 Port

```
USB2_PHY_PARAMETERS USB2_PHY_CONFIG::Port[MAX_USB2_PORTS]
```

This structure configures per USB2 port physical settings.

It allows to setup the port location and port length, and configures the port strength accordingly. Changing this policy values from default ones may require disabling USB2 PHY Sus Well Power Gating through Usb2PhySusPgEnable on PCH-LP

Definition at line 110 of file Usb2PhyConfig.h.

The documentation for this struct was generated from the following file:

- Usb2PhyConfig.h

12.156 USB2_PHY_PARAMETERS Struct Reference

This structure configures per USB2 AFE settings.

```
#include <Usb2PhyConfig.h>
```

Public Attributes

- UINT8 Petxiset

  Per Port HS Preemphasis Bias (PERPORTPETXISET) 000b - 0mV 001b - 11.25mV 010b - 16.9mV 011b - 28.15mV 100b - 28.15mV 101b - 39.35mV 110b - 45mV 111b - 56.3mV.
- UINT8 Txiset

  Per Port HS Transmitter Bias (PERPORTTXISET) 000b - 0mV 001b - 11.25mV 010b - 16.9mV 011b - 28.15mV 100b - 28.15mV 101b - 39.35mV 110b - 45mV 111b - 56.3mV.
- UINT8 Predeemp

  Per Port HS Transmitter Emphasis (IUSBTXEMPHASISEN) 00b - Emphasis OFF 01b - De-emphasis ON 10b - Pre-emphasis ON 11b - Pre-emphasis & De-emphasis ON.
- UINT8 Pehalfbit

  Per Port Half Bit Pre-emphasis (PERPORTTXPEHALF) 1b - half-bit pre-emphasis 0b - full-bit pre-emphasis.
- UINT8 PredeempSemiflexEn

  Semi Flexi Pre/DE-emphasis (reg_predeemp_semiflexi_en): 1: Choose the predeemp setting (increase the DC amplitude) 0: Choose the predeemp setting (decrease the DC amplitude)

12.156.1 Detailed Description

This structure configures per USB2 AFE settings.

It allows to setup the port electrical parameters.

Definition at line 49 of file Usb2PhyConfig.h.

The documentation for this struct was generated from the following file:

- Usb2PhyConfig.h

12.157 USB2_PORT_CONFIG Struct Reference

This structure configures per USB2.0 port settings like enabling and overcurrent protection.

```
#include <UsbConfig.h>
```

Public Attributes

- UINT32 OverCurrentPin: 8

  These members describe the specific over current pin number of USB 2.0 Port N.
- UINT32 Enable: 1

  *0: Disable; **1: Enable**.*
- UINT32 PortResetMessageEnable: 1

  0: Disable USB2 Port Reset Message; 1: Enable USB2 Port Reset Message
- UINT32 SwDeviceModeEnable: 1

  0: Disable USB2 Port Device Mode; 1: Force USB2 Port Device Mode
- UINT32 RsvdBits0: 21

  Reserved bits.

12.157.1 Detailed Description

This structure configures per USB2.0 port settings like enabling and overcurrent protection.

Definition at line 60 of file UsbConfig.h.

12.157.2 Member Data Documentation

12.157.2.1 OverCurrentPin

```
UINT32 USB2_PORT_CONFIG::OverCurrentPin
```

These members describe the specific over current pin number of USB 2.0 Port N.

It is SW's responsibility to ensure that a given port's bit map is set only for one OC pin Description. USB2 and USB3 on the same combo Port must use the same OC pin.

Definition at line 66 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

- UsbConfig.h

12.158 USB3_PORT_CONFIG Struct Reference

This structure configures per USB3.x port settings like enabling and overcurrent protection.

```
#include <UsbConfig.h>
```

Public Attributes

- UINT32 OverCurrentPin: 8
  These members describe the specific over current pin number of USB 3.x Port N.
- UINT32 Enable: 1
  *0: Disable; **1: Enable**.*
- UINT32 RsvdBits0: 23
  Reserved bits.

12.158.1 Detailed Description

This structure configures per USB3.x port settings like enabling and overcurrent protection.

Definition at line 76 of file UsbConfig.h.

12.158.2 Member Data Documentation

12.158.2.1 OverCurrentPin

```
UINT32 USB3_PORT_CONFIG::OverCurrentPin
```

These members describe the specific over current pin number of USB 3.x Port N.

It is SW's responsibility to ensure that a given port's bit map is set only for one OC pin Description. USB2 and USB3 on the same combo Port must use the same OC pin.

Definition at line 82 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

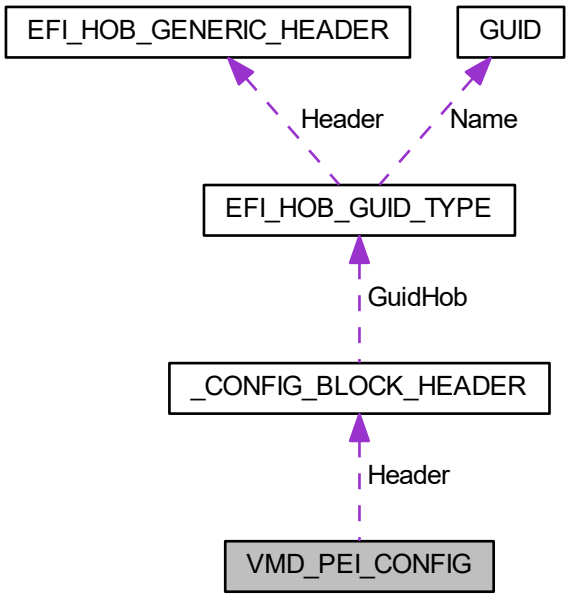
- UsbConfig.h

12.159 VMD_PEI_CONFIG Struct Reference

This configuration block is to configure VMD related variables used in PostMem PEI.

```
#include <VmdPeiConfig.h>
```

Collaboration diagram for VMD_PEI_CONFIG:



Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Config Block Header.
- UINT8 VmdEnable

  *Offset 28 This field used to enable VMD controller **1=Enable(default)** 0=Disable.*
- UINT8 VmdGlobalMapping

  *Offset 29 This field used to enable Global Mapping 1=Enable **0=Disable(default)***
- UINT8 Rsvd [2]

  Offset 30-39 Reserved.
- RP_BDF_DATA VmdPortEnable [VMD_MAX_DEVICES]

  *Offset 40 to 163 This field used to to store b/d/f for each root port along with enable Support 1=Enable **0=Disable (default)***
- VOID * VmdVariablePtr

  This config block will be updated as per the EFI variable.
- UINT32 VmdCfgBarBase

  *Temp Address VMD CFG BAR Default is **0xA0000000***
- UINT32 VmdMemBar1Base

  *Temp Address VMD CFG BAR Default is **0xA2000000***
- UINT32 VmdMemBar2Base

  *Temp Address VMD CFG BAR Default is **0xA4000000***

12.159.1 Detailed Description

This configuration block is to configure VMD related variables used in PostMem PEI.

If VMD Device is not supported, all policies can be ignored. **Revision 1**:

- Initial version.

Definition at line 61 of file VmdPeiConfig.h.

The documentation for this struct was generated from the following file:

- VmdPeiConfig.h

12.160 VTD_ENGINE_CONFIG Struct Reference

This structure describes each VT-d engine.

`#include <VtdConfig.h>`

12.160.1 Detailed Description

This structure describes each VT-d engine.

Definition at line 80 of file VtdConfig.h.

The documentation for this struct was generated from the following file:

- VtdConfig.h

12.161 XDCI_CONFIG Struct Reference

The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller.

`#include <UsbConfig.h>`

Public Attributes

- UINT32 Enable: 1

  This member describes whether or not the xDCI controller should be enabled.
- UINT32 RsvdBits0: 31

  Reserved bits.

12.161.1 Detailed Description

The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller.

Definition at line 91 of file UsbConfig.h.

12.161.2 Member Data Documentation

12.161.2.1 Enable

`UINT32 XDCI_CONFIG::Enable`

This member describes whether or not the xDCI controller should be enabled.

0: Disable; **1: Enable**.

Definition at line 96 of file UsbConfig.h.

The documentation for this struct was generated from the following file:

- UsbConfig.h

Chapter 13

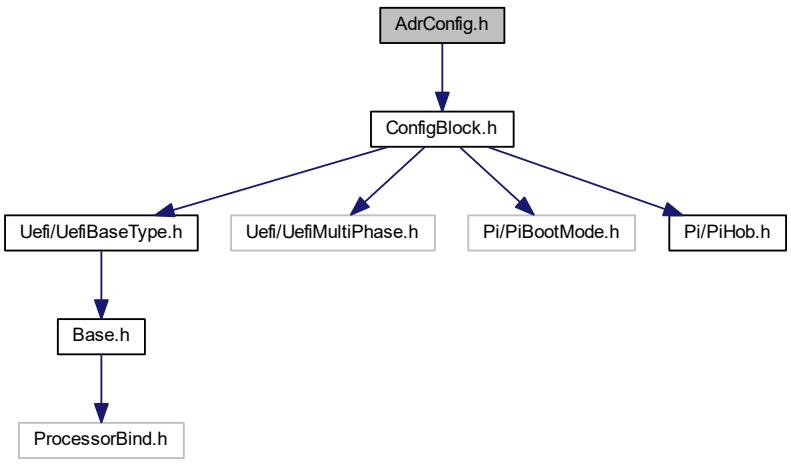
File Documentation

13.1 AdrConfig.h File Reference

ADR policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for AdrConfig.h:



Classes

- union ADR_GLOBAL_RESET_ENABLE

  ADR Source Enable.
- struct ADR_CONFIG

  ADR Configuration ***Revision 1****: - Initial version.*

13.1.1 Detailed Description

ADR policy.

Copyright

INTEL CONFIDENTIAL Copyright 2019 - 2020 Intel Corporation.

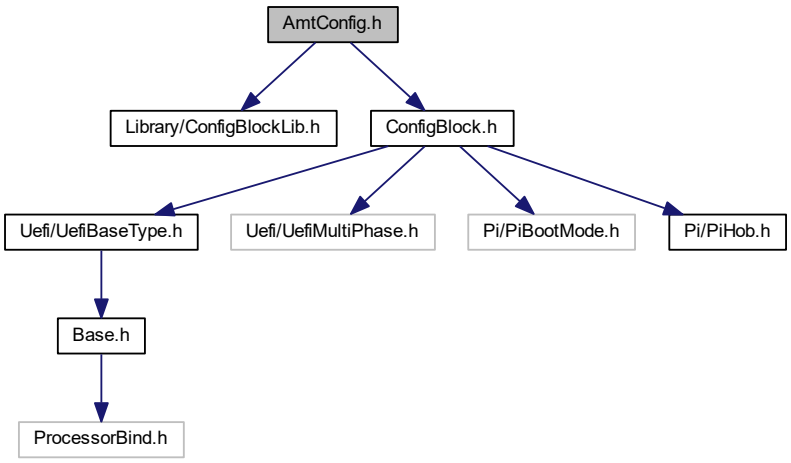
Specification Reference:

13.2 AmtConfig.h File Reference

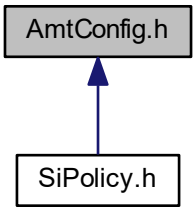
AMT Config Block for PEI/DXE phase.

```
#include <Library/ConfigBlockLib.h>
#include <ConfigBlock.h>
```

Include dependency graph for AmtConfig.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct AMT_PEI_CONFIG
  AMT Pei Configuration Structure.
- struct AMT_DXE_CONFIG
  AMT Dxe Configuration Structure.

Typedefs

- typedef VOID(∗ AMT_REPORT_ERROR) (IN AMT_ERROR_MSG_ID MsgId)
  Show AMT Error message.

Enumerations

- enum AMT_ERROR_MSG_ID
  AMT Error Message ID.

13.2.1 Detailed Description

AMT Config Block for PEI/DXE phase.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2020 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.2.2 Typedef Documentation

13.2.2.1 AMT_REPORT_ERROR

```
typedef VOID(* AMT_REPORT_ERROR) (IN AMT_ERROR_MSG_ID MsgId)
```

Show AMT Error message.

This is to display localized message in the console. This is used to display message strings in local language. To display the message, the routine will check the message ID and ConOut the message strings. For example, the End of Post error displayed in English will be: gST->ConOut->OutputString (gST->ConOut, L"Error sending End Of Post message to ME\n"); It is recommended to clear the screen before displaying the error message and keep the message on the screen for several seconds. A sample is provided, see ShowAmtReportError () to retrieve details.

Parameters

in	*MsgId*	AMT error message ID for displaying on screen message

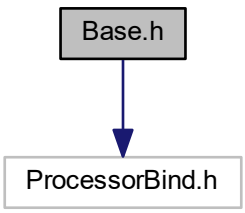
Definition at line 75 of file AmtConfig.h.

13.3 Base.h File Reference

Root include file for Mde Package Base type modules.

```
#include <ProcessorBind.h>
```

Include dependency graph for Base.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct GUID

  128 bit buffer containing a unique identifier value.
- struct IPv4_ADDRESS

  4-byte buffer.
- struct IPv6_ADDRESS

  16-byte buffer.
- struct _LIST_ENTRY

  _LIST_ENTRY structure definition.

Macros

- #define GLOBAL_REMOVE_IF_UNREFERENCED

  Remove the global variable from the linked image if there are no references to it after all compiler and linker optimizations have been performed.
- #define UNREACHABLE()

  Signal compilers and analyzers that this call is not reachable.
- #define NORETURN

  Signal compilers and analyzers that the function cannot return.
- #define ANALYZER_UNREACHABLE()

  Signal the analyzer that this call is not reachable.
- #define ANALYZER_NORETURN

  Signal the analyzer that the function cannot return.
- #define RETURNS_TWICE

  Tell the code optimizer that the function will return twice.
- #define _CONCATENATE(a, b) __CONCATENATE(a, b)

  Private worker functions for ASM_PFX()
- #define ASM_PFX(name) _CONCATENATE (__USER_LABEL_PREFIX__, name)

  *The **USER_LABEL_PREFIX** macro predefined by GNUC represents the prefix on symbols in assembly language.*
- #define CONST const

  Datum is read-only.
- #define STATIC static

  Datum is scoped to the current file or function.
- #define VOID void

  Undeclared type.
- #define IN

  Datum is passed to the function.
- #define OUT

  Datum is returned from the function.
- #define OPTIONAL

  Passing the datum to the function is optional, and a NULL is passed if the value is not supplied.

- #define TRUE ((BOOLEAN)(1==1))

  Boolean true value.
- #define FALSE ((BOOLEAN)(0==1))

  Boolean false value.
- #define NULL ((VOID *) 0)

  *NULL pointer (VOID *)*
- #define MAX_INT8 ((INT8)0x7F)

  Maximum values for common UEFI Data Types.
- #define MIN_INT8 (((INT8) -127) - 1)

  Minimum values for the signed UEFI Data Types.
- #define _INT_SIZE_OF(n) ((sizeof (n) + sizeof (UINTN) - 1) &~(sizeof (UINTN) - 1))

  Return the size of argument that has been aligned to sizeof (UINTN).
- #define VA_START(Marker, Parameter) (Marker = (VA_LIST) ((UINTN) & (Parameter) + _INT_SIZE_OF (Parameter)))

  Retrieves a pointer to the beginning of a variable argument list, based on the name of the parameter that immediately precedes the variable argument list.
- #define VA_ARG(Marker, TYPE) (*(TYPE *) ((Marker += _INT_SIZE_OF (TYPE)) - _INT_SIZE_OF (TYPE)))

  Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.
- #define VA_END(Marker) (Marker = (VA_LIST) 0)

  Terminates the use of a variable argument list.
- #define VA_COPY(Dest, Start) ((void)((Dest) = (Start)))

  Initializes a VA_LIST as a copy of an existing VA_LIST.
- #define _BASE_INT_SIZE_OF(TYPE) ((sizeof (TYPE) + sizeof (UINTN) - 1) / sizeof (UINTN))

  Returns the size of a data type in sizeof(UINTN) units rounded up to the nearest UINTN boundary.
- #define BASE_ARG(Marker, TYPE) (*(TYPE *) ((Marker += _BASE_INT_SIZE_OF (TYPE)) - _BASE_INT_SIZE_OF (TYPE)))

  Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.
- #define OFFSET_OF(TYPE, Field) ((UINTN) &(((TYPE *)0)->Field))

  The macro that returns the byte offset of a field in a data structure.
- #define ALIGNOF(TYPE) OFFSET_OF (struct { CHAR8 C; TYPE A; }, A)

  Returns the alignment requirement of a type.
- #define STATIC_ASSERT _Static_assert

  Portable definition for compile time assertions.
- #define BASE_CR(Record, TYPE, Field) ((TYPE *) ((CHAR8 *) (Record) - OFFSET_OF (TYPE, Field)))

  Macro that returns a pointer to the data structure that contains a specified field of that data structure.
- #define IS_POW2(Value) ((Value) != 0U && ((Value) & ((Value) - 1U)) == 0U)

  Checks whether a value is a power of two.
- #define IS_ALIGNED(Value, Alignment) (((Value) & ((Alignment) - 1U)) == 0U)

  Checks whether a value is aligned by a specified alignment.
- #define ADDRESS_IS_ALIGNED(Address, Alignment) IS_ALIGNED ((UINTN) (Address), Alignment)

  Checks whether a pointer or address is aligned by a specified alignment.
- #define ALIGN_VALUE_ADDEND(Value, Alignment) (((Alignment) - (Value)) & ((Alignment) - 1U))

  Determines the addend to add to a value to round it up to the next boundary of a specified alignment.
- #define ALIGN_VALUE(Value, Alignment) ((Value) + ALIGN_VALUE_ADDEND (Value, Alignment))

  Rounds a value up to the next boundary using a specified alignment.
- #define ALIGN_POINTER(Pointer, Alignment) ((VOID *) (ALIGN_VALUE ((UINTN)(Pointer), (Alignment))))

  Adjust a pointer by adding the minimum offset required for it to be aligned on a specified alignment boundary.
- #define ALIGN_VARIABLE(Value) ALIGN_VALUE ((Value), sizeof (UINTN))

  Rounds a value up to the next natural boundary for the current CPU.

- #define MAX(a, b) (((a) > (b)) ? (a) : (b))

  Return the maximum of two operands.
- #define MIN(a, b) (((a) < (b)) ? (a) : (b))

  Return the minimum of two operands.
- #define ABS(a) (((a) < 0) ? (-(a)) : (a))

  Return the absolute value of a signed operand.
- #define ENCODE_ERROR(StatusCode) ((RETURN_STATUS)(MAX_BIT | (StatusCode)))

  Produces a RETURN_STATUS code with the highest bit set.
- #define ENCODE_WARNING(StatusCode) ((RETURN_STATUS)(StatusCode))

  Produces a RETURN_STATUS code with the highest bit clear.
- #define RETURN_ERROR(StatusCode) (((INTN)(RETURN_STATUS)(StatusCode)) < 0)

  Returns TRUE if a specified RETURN_STATUS code is an error code.
- #define RETURN_SUCCESS (RETURN_STATUS)(0)

  The operation completed successfully.
- #define RETURN_LOAD_ERROR ENCODE_ERROR (1)

  The image failed to load.
- #define RETURN_INVALID_PARAMETER ENCODE_ERROR (2)

  The parameter was incorrect.
- #define RETURN_UNSUPPORTED ENCODE_ERROR (3)

  The operation is not supported.
- #define RETURN_BAD_BUFFER_SIZE ENCODE_ERROR (4)

  The buffer was not the proper size for the request.
- #define RETURN_BUFFER_TOO_SMALL ENCODE_ERROR (5)

  The buffer was not large enough to hold the requested data.
- #define RETURN_NOT_READY ENCODE_ERROR (6)

  There is no data pending upon return.
- #define RETURN_DEVICE_ERROR ENCODE_ERROR (7)

  The physical device reported an error while attempting the operation.
- #define RETURN_WRITE_PROTECTED ENCODE_ERROR (8)

  The device can not be written to.
- #define RETURN_OUT_OF_RESOURCES ENCODE_ERROR (9)

  The resource has run out.
- #define RETURN_VOLUME_CORRUPTED ENCODE_ERROR (10)

  An inconsistency was detected on the file system causing the operation to fail.
- #define RETURN_VOLUME_FULL ENCODE_ERROR (11)

  There is no more space on the file system.
- #define RETURN_NO_MEDIA ENCODE_ERROR (12)

  The device does not contain any medium to perform the operation.
- #define RETURN_MEDIA_CHANGED ENCODE_ERROR (13)

  The medium in the device has changed since the last access.
- #define RETURN_NOT_FOUND ENCODE_ERROR (14)

  The item was not found.
- #define RETURN_ACCESS_DENIED ENCODE_ERROR (15)

  Access was denied.
- #define RETURN_NO_RESPONSE ENCODE_ERROR (16)

  The server was not found or did not respond to the request.
- #define RETURN_NO_MAPPING ENCODE_ERROR (17)

  A mapping to the device does not exist.
- #define RETURN_TIMEOUT ENCODE_ERROR (18)

  A timeout time expired.
- #define RETURN_NOT_STARTED ENCODE_ERROR (19)

The protocol has not been started.

- #define RETURN_ALREADY_STARTED ENCODE_ERROR (20)

  The protocol has already been started.
- #define RETURN_ABORTED ENCODE_ERROR (21)

  The operation was aborted.
- #define RETURN_ICMP_ERROR ENCODE_ERROR (22)

  An ICMP error occurred during the network operation.
- #define RETURN_TFTP_ERROR ENCODE_ERROR (23)

  A TFTP error occurred during the network operation.
- #define RETURN_PROTOCOL_ERROR ENCODE_ERROR (24)

  A protocol error occurred during the network operation.
- #define RETURN_INCOMPATIBLE_VERSION ENCODE_ERROR (25)

  A function encountered an internal version that was incompatible with a version requested by the caller.
- #define RETURN_SECURITY_VIOLATION ENCODE_ERROR (26)

  The function was not performed due to a security violation.
- #define RETURN_CRC_ERROR ENCODE_ERROR (27)

  A CRC error was detected.
- #define RETURN_END_OF_MEDIA ENCODE_ERROR (28)

  The beginning or end of media was reached.
- #define RETURN_END_OF_FILE ENCODE_ERROR (31)

  The end of the file was reached.
- #define RETURN_INVALID_LANGUAGE ENCODE_ERROR (32)

  The language specified was invalid.
- #define RETURN_COMPROMISED_DATA ENCODE_ERROR (33)

  The security status of the data is unknown or compromised and the data must be updated or replaced to restore a valid security status.
- #define RETURN_HTTP_ERROR ENCODE_ERROR (35)

  A HTTP error occurred during the network operation.
- #define RETURN_WARN_UNKNOWN_GLYPH ENCODE_WARNING (1)

  The string contained one or more characters that the device could not render and were skipped.
- #define RETURN_WARN_DELETE_FAILURE ENCODE_WARNING (2)

  The handle was closed, but the file was not deleted.
- #define RETURN_WARN_WRITE_FAILURE ENCODE_WARNING (3)

  The handle was closed, but the data to the file was not flushed properly.
- #define RETURN_WARN_BUFFER_TOO_SMALL ENCODE_WARNING (4)

  The resulting buffer was too small, and the data was truncated to the buffer size.
- #define RETURN_WARN_STALE_DATA ENCODE_WARNING (5)

  The data has not been updated within the timeframe set by local policy for this type of data.
- #define RETURN_WARN_FILE_SYSTEM ENCODE_WARNING (6)

  The resulting buffer contains UEFI-compliant file system.
- #define SIGNATURE_16(A, B) ((A) | (B << 8))

  Returns a 16-bit signature built from 2 ASCII characters.
- #define SIGNATURE_32(A, B, C, D) (SIGNATURE_16 (A, B) | (SIGNATURE_16 (C, D) << 16))

  Returns a 32-bit signature built from 4 ASCII characters.
- #define SIGNATURE_64(A, B, C, D, E, F, G, H) (SIGNATURE_32 (A, B, C, D) | ((UINT64) (SIGNATURE_32 (E, F, G, H)) << 32))

  Returns a 64-bit signature built from 8 ASCII characters.
- #define RETURN_ADDRESS(L) ((VOID *) 0)

  Get the return address of the calling function.
- #define ARRAY_SIZE(Array) (sizeof (Array) / sizeof ((Array)[0]))

  Return the number of elements in an array.

Typedefs

- typedef struct _LIST_ENTRY LIST_ENTRY

  LIST_ENTRY structure definition.
- typedef CHAR8 * VA_LIST

  Variable used to traverse the list of arguments.
- typedef UINTN * BASE_LIST

  Pointer to the start of a variable argument list stored in a memory buffer.

13.3.1 Detailed Description

Root include file for Mde Package Base type modules.

This is the include file for any module of type base. Base modules only use types defined via this include file and can be ported easily to any environment. There are a set of base libraries in the Mde Package that can be used to implement base modules.

13.3.2 Macro Definition Documentation

13.3.2.1 _BASE_INT_SIZE_OF

```
#define _BASE_INT_SIZE_OF(
              TYPE ) ((sizeof (TYPE) + sizeof (UINTN) - 1) / sizeof (UINTN))
```

Returns the size of a data type in sizeof(UINTN) units rounded up to the nearest UINTN boundary.

Parameters

TYPE	The date type to determine the size of.

Returns

The size of TYPE in sizeof (UINTN) units rounded up to the nearest UINTN boundary.

Definition at line 720 of file Base.h.

13.3.2.2 _INT_SIZE_OF

```
#define _INT_SIZE_OF(
              n ) ((sizeof (n) + sizeof (UINTN) - 1) &~(sizeof (UINTN) - 1))
```

Return the size of argument that has been aligned to sizeof (UINTN).

Parameters

n	The parameter size to be aligned.

Returns

The aligned size.

Definition at line 579 of file Base.h.

13.3.2.3 ABS

```
#define ABS(
              a )  (((a) < 0) ?  (-(a)) :  (a))
```

Return the absolute value of a signed operand.

This macro returns the absolute value of the signed operand specified by a.

Parameters

a	The signed operand.

Returns

The absolute value of the signed operand.

Definition at line 1020 of file Base.h.

13.3.2.4 ADDRESS_IS_ALIGNED

```
#define ADDRESS_IS_ALIGNED(
              Address,
              Alignment ) IS_ALIGNED ((UINTN) (Address), Alignment)
```

Checks whether a pointer or address is aligned by a specified alignment.

Parameters

Address	The pointer or address to check.
Alignment	The alignment boundary used to check against.

Return values

TRUE	Address is aligned by Alignment.

Return values

FALSE	Address is not aligned by Alignment.

Definition at line 923 of file Base.h.

13.3.2.5 ALIGN_POINTER

```
#define ALIGN_POINTER(
              Pointer,
              Alignment ) ((VOID *) (ALIGN_VALUE ((UINTN)(Pointer), (Alignment))))
```

Adjust a pointer by adding the minimum offset required for it to be aligned on a specified alignment boundary.

This function rounds the pointer specified by Pointer to the next alignment boundary specified by Alignment. The pointer to the aligned address is returned.

Parameters

Pointer	The pointer to round up.
Alignment	The alignment boundary to use to return an aligned pointer.

Returns

Pointer to the aligned address.

Definition at line 963 of file Base.h.

13.3.2.6 ALIGN_VALUE

```
#define ALIGN_VALUE(
              Value,
              Alignment ) ((Value) + ALIGN_VALUE_ADDEND (Value, Alignment))
```

Rounds a value up to the next boundary using a specified alignment.

This function rounds Value up to the next boundary using the specified Alignment. This aligned value is returned.

Parameters

Value	The value to round up.
Alignment	The alignment boundary used to return the aligned value.

Returns

A value up to the next boundary.

Definition at line 948 of file Base.h.

13.3.2.7 ALIGN_VALUE_ADDEND

```
#define ALIGN_VALUE_ADDEND(
              Value,
              Alignment ) (((Alignment) - (Value)) & ((Alignment) - 1U))
```

Determines the addend to add to a value to round it up to the next boundary of a specified alignment.

Parameters

Value	The value to round up.
Alignment	The alignment boundary used to return the addend.

Returns

Addend to round Value up to alignment boundary Alignment.

Definition at line 934 of file Base.h.

13.3.2.8 ALIGN_VARIABLE

```
#define ALIGN_VARIABLE(
              Value ) ALIGN_VALUE ((Value), sizeof (UINTN))
```

Rounds a value up to the next natural boundary for the current CPU.

This is 4-bytes for 32-bit CPUs and 8-bytes for 64-bit CPUs.

This function rounds the value specified by Value up to the next natural boundary for the current CPU. This rounded value is returned.

Parameters

Value	The value to round up.

Returns

Rounded value specified by Value.

Definition at line 977 of file Base.h.

13.3.2.9 ALIGNOF

```
#define ALIGNOF(
              TYPE ) OFFSET_OF (struct { CHAR8 C; TYPE A; }, A)
```

Returns the alignment requirement of a type.

Parameters

TYPE	The name of the type to retrieve the alignment requirement of.

Returns

Alignment requirement, in Bytes, of TYPE.

Definition at line 792 of file Base.h.

13.3.2.10 ANALYZER_NORETURN

```
#define ANALYZER_NORETURN
```

Signal the analyzer that the function cannot return.

This excludes compilers.

Definition at line 157 of file Base.h.

13.3.2.11 ANALYZER_UNREACHABLE

```
#define ANALYZER_UNREACHABLE( )
```

Signal the analyzer that this call is not reachable.

This excludes compilers.

Definition at line 131 of file Base.h.

13.3.2.12 ARRAY_SIZE

```
#define ARRAY_SIZE(
              Array ) (sizeof (Array) / sizeof ((Array)[0]))
```

Return the number of elements in an array.

Parameters

Array	An object of array type. Array is only used as an argument to the sizeof operator, therefore Array is never evaluated. The caller is responsible for ensuring that Array's type is not incomplete; that is, Array must have known constant size.

Returns

The number of elements in Array. The result has type UINTN.

Definition at line 1383 of file Base.h.

13.3.2.13 BASE_ARG

```
#define BASE_ARG(
              Marker,
              TYPE ) (*(TYPE *) ((Marker += _BASE_INT_SIZE_OF (TYPE)) - _BASE_INT_SIZE_OF (TY↩
PE)))
```

Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.

This function returns an argument of the type specified by TYPE from the beginning of the variable argument list specified by Marker. Marker is then updated to point to the next argument in the variable argument list. The method for computing the pointer to the next argument in the argument list is CPU specific following the EFIAPI ABI.

Parameters

Marker	The pointer to the beginning of a variable argument list.
TYPE	The type of argument to retrieve from the beginning of the variable argument list.

Returns

An argument of the type specified by TYPE.

Definition at line 738 of file Base.h.

13.3.2.14 BASE_CR

```
#define BASE_CR(
              Record,
              TYPE,
              Field ) ((TYPE *) ((CHAR8 *) (Record) - OFFSET_OF (TYPE, Field)))
```

Macro that returns a pointer to the data structure that contains a specified field of that data structure.

This is a lightweight method to hide information by placing a public data structure inside a larger private data structure and using a pointer to the public data structure to retrieve a pointer to the private data structure.

This function computes the offset, in bytes, of field specified by Field from the beginning of the data structure specified by TYPE. This offset is subtracted from Record, and is used to return a pointer to a data structure of the type specified by TYPE. If the data type specified by TYPE does not contain the field specified by Field, then the module will not compile.

Parameters

Record	Pointer to the field specified by Field within a data structure of type TYPE.
TYPE	The name of the data structure type to return. This data structure must contain the field specified by Field.
Field	The name of the field in the data structure specified by TYPE to which Record points.

Returns

A pointer to the structure from one of it's elements.

Definition at line 891 of file Base.h.

13.3.2.15 ENCODE_ERROR

```
#define ENCODE_ERROR(
              StatusCode ) ((RETURN_STATUS)(MAX_BIT | (StatusCode)))
```

Produces a RETURN_STATUS code with the highest bit set.

Parameters

StatusCode	The status code value to convert into a warning code. StatusCode must be in the range 0x00000000..0x7FFFFFFF.

Returns

The value specified by StatusCode with the highest bit set.

Definition at line 1037 of file Base.h.

13.3.2.16 ENCODE_WARNING

```
#define ENCODE_WARNING(
              StatusCode ) ((RETURN_STATUS)(StatusCode))
```

Produces a RETURN_STATUS code with the highest bit clear.

Parameters

StatusCode	The status code value to convert into a warning code. StatusCode must be in the range 0x00000000..0x7FFFFFFF.

Returns

The value specified by StatusCode with the highest bit clear.

Definition at line 1048 of file Base.h.

13.3.2.17 FALSE

```
#define FALSE ((BOOLEAN)(0==1))
```

Boolean false value.

UEFI Specification defines this value to be 0, but this form is more portable.

Definition at line 307 of file Base.h.

13.3.2.18 IS_ALIGNED

```
#define IS_ALIGNED(
              Value,
              Alignment ) (((Value) & ((Alignment) - 1U)) == 0U)
```

Checks whether a value is aligned by a specified alignment.

Parameters

Value	The value to check.
Alignment	The alignment boundary used to check against.

Return values

TRUE	Value is aligned by Alignment.
FALSE	Value is not aligned by Alignment.

Definition at line 912 of file Base.h.

13.3.2.19 IS_POW2

```
#define IS_POW2(
              Value ) ((Value) != 0U && ((Value) & ((Value) - 1U)) == 0U)
```

Checks whether a value is a power of two.

Parameters

Value	The value to check.

Return values

TRUE	Value is a power of two.
FALSE	Value is not a power of two.

Definition at line 901 of file Base.h.

13.3.2.20 MAX

```
#define MAX(
              a,
              b )  (((a) > (b)) ?  (a) :  (b))
```

Return the maximum of two operands.

This macro returns the maximum of two operand specified by a and b. Both a and b must be the same numerical types, signed or unsigned.

Parameters

a	The first operand with any numerical type.
b	The second operand. Can be any numerical type as long as is the same type as a.

Returns

Maximum of two operands.

Definition at line 992 of file Base.h.

13.3.2.21 MIN

```
#define MIN(
              a,
              b )  (((a) < (b)) ?  (a) :  (b))
```

Return the minimum of two operands.

This macro returns the minimal of two operand specified by a and b. Both a and b must be the same numerical types, signed or unsigned.

Parameters

a	The first operand with any numerical type.
b	The second operand. It should be the same any numerical type with a.

Returns

Minimum of two operands.

Definition at line 1007 of file Base.h.

13.3.2.22 NORETURN

```
#define NORETURN
```

Signal compilers and analyzers that the function cannot return.

It is up to the compiler to remove any code past a call to functions flagged with this attribute.

Definition at line 107 of file Base.h.

13.3.2.23 OFFSET_OF

```
#define OFFSET_OF(
              TYPE,
              Field ) ((UINTN) &(((TYPE *)0)->Field))
```

The macro that returns the byte offset of a field in a data structure.

This function returns the offset, in bytes, of field specified by Field from the beginning of the data structure specified by TYPE. If TYPE does not contain Field, the module will not compile.

Parameters

TYPE	The name of the data structure that contains the field specified by Field.
Field	The name of the field in the data structure.

Returns

Offset, in bytes, of field.

Definition at line 758 of file Base.h.

13.3.2.24 RETURN_ADDRESS

```
#define RETURN_ADDRESS(
              L ) ((VOID *) 0)
```

Get the return address of the calling function.

Parameters

L	Return Level.

Returns

0 as compilers don't support this feature.

Definition at line 1369 of file Base.h.

13.3.2.25 RETURN_BUFFER_TOO_SMALL

```
#define RETURN_BUFFER_TOO_SMALL ENCODE_ERROR (5)
```

The buffer was not large enough to hold the requested data.

The required buffer size is returned in the appropriate parameter when this error occurs.

Definition at line 1093 of file Base.h.

13.3.2.26 RETURN_ERROR

```
#define RETURN_ERROR(
              StatusCode ) (((INTN)(RETURN_STATUS)(StatusCode)) < 0)
```

Returns TRUE if a specified RETURN_STATUS code is an error code.

This function returns TRUE if StatusCode has the high bit set. Otherwise, FALSE is returned.

Parameters

StatusCode	The status code value to evaluate.

Return values

TRUE	The high bit of StatusCode is set.
FALSE	The high bit of StatusCode is clear.

Definition at line 1061 of file Base.h.

13.3.2.27 RETURNS_TWICE

```
#define RETURNS_TWICE
```

Tell the code optimizer that the function will return twice.

This prevents wrong optimizations which can cause bugs. Tell the code optimizer that the function will return twice. This prevents wrong optimizations which can cause bugs.

Definition at line 177 of file Base.h.

13.3.2.28 SIGNATURE_16

```
#define SIGNATURE_16(
              A,
              B ) ((A) | (B << 8))
```

Returns a 16-bit signature built from 2 ASCII characters.

This macro returns a 16-bit value built from the two ASCII characters specified by A and B.

Parameters

A	The first ASCII character.
B	The second ASCII character.

Returns

A 16-bit value built from the two ASCII characters specified by A and B.

Definition at line 1283 of file Base.h.

13.3.2.29 SIGNATURE_32

```
#define SIGNATURE_32(
              A,
              B,
              C,
              D ) (SIGNATURE_16 (A, B) | (SIGNATURE_16 (C, D) << 16))
```

Returns a 32-bit signature built from 4 ASCII characters.

This macro returns a 32-bit value built from the four ASCII characters specified by A, B, C, and D.

Parameters

A	The first ASCII character.
B	The second ASCII character.
C	The third ASCII character.
D	The fourth ASCII character.

Returns

A 32-bit value built from the two ASCII characters specified by A, B, C and D.

Definition at line 1300 of file Base.h.

13.3.2.30 SIGNATURE_64

```
#define SIGNATURE_64(
              A,
              B,
              C,
              D,
              E,
              F,
              G,
              H )  (SIGNATURE_32 (A, B, C, D) | ((UINT64) (SIGNATURE_32 (E, F, G, H)) << 32))
```

Returns a 64-bit signature built from 8 ASCII characters.

This macro returns a 64-bit value built from the eight ASCII characters specified by A, B, C, D, E, F, G,and H.

Parameters

A	The first ASCII character.
B	The second ASCII character.
C	The third ASCII character.
D	The fourth ASCII character.
E	The fifth ASCII character.
F	The sixth ASCII character.
G	The seventh ASCII character.
H	The eighth ASCII character.

Returns

A 64-bit value built from the two ASCII characters specified by A, B, C, D, E, F, G and H.

Definition at line 1321 of file Base.h.

13.3.2.31 STATIC_ASSERT

```
#define STATIC_ASSERT _Static_assert
```

Portable definition for compile time assertions.

Equivalent to C11 static_assert macro from assert.h.

Parameters

Expression	Boolean expression.
Message	Raised compiler diagnostic message when expression is false.

Definition at line 808 of file Base.h.

13.3.2.32 TRUE

```
#define TRUE ((BOOLEAN)(1==1))
```

Boolean true value.

UEFI Specification defines this value to be 1, but this form is more portable.

Definition at line 301 of file Base.h.

13.3.2.33 UNREACHABLE

```
#define UNREACHABLE( )
```

Signal compilers and analyzers that this call is not reachable.

It is up to the compiler to remove any code past that point.

Definition at line 77 of file Base.h.

13.3.2.34 VA_ARG

```
#define VA_ARG(
              Marker,
              TYPE ) (*(TYPE *) ((Marker += _INT_SIZE_OF (TYPE)) - _INT_SIZE_OF (TYPE)))
```

Returns an argument of a specified type from a variable argument list and updates the pointer to the variable argument list to point to the next argument.

This function returns an argument of the type specified by TYPE from the beginning of the variable argument list specified by Marker. Marker is then updated to point to the next argument in the variable argument list. The method for computing the pointer to the next argument in the argument list is CPU-specific following the EFIAPI ABI.

Parameters

Marker	VA_LIST used to traverse the list of arguments.
TYPE	The type of argument to retrieve from the beginning of the variable argument list.

Returns

An argument of the type specified by TYPE.

Definition at line 679 of file Base.h.

13.3.2.35 VA_COPY

```
#define VA_COPY(
              Dest,
              Start ) ((void)((Dest) = (Start)))
```

Initializes a VA_LIST as a copy of an existing VA_LIST.

This macro initializes Dest as a copy of Start, as if the VA_START macro had been applied to Dest followed by the same sequence of uses of the VA_ARG macro as had previously been used to reach the present state of Start.

Parameters

Dest	VA_LIST used to traverse the list of arguments.
Start	VA_LIST used to traverse the list of arguments.

Definition at line 704 of file Base.h.

13.3.2.36 VA_END

```
#define VA_END(
              Marker ) (Marker = (VA_LIST) 0)
```

Terminates the use of a variable argument list.

This function initializes Marker so it can no longer be used with VA_ARG(). After this macro is used, the only way to access the variable argument list is by using VA_START() again.

Parameters

Marker	VA_LIST used to traverse the list of arguments.

Definition at line 691 of file Base.h.

13.3.2.37 VA_START

```
#define VA_START(
              Marker,
              Parameter ) (Marker = (VA_LIST) ((UINTN) & (Parameter) + _INT_SIZE_OF (Parameter)))
```

Retrieves a pointer to the beginning of a variable argument list, based on the name of the parameter that immediately precedes the variable argument list.

This function initializes Marker to point to the beginning of the variable argument list that immediately follows Parameter. The method for computing the pointer to the next argument in the argument list is CPU-specific following the EFIAPI ABI.

Parameters

Marker	The VA_LIST used to traverse the list of arguments.
Parameter	The name of the parameter that immediately precedes the variable argument list.

Returns

A pointer to the beginning of a variable argument list.

Definition at line 661 of file Base.h.

13.3.3 Typedef Documentation

13.3.3.1 BASE_LIST

```
typedef UINTN* BASE_LIST
```

Pointer to the start of a variable argument list stored in a memory buffer.

Same as UINT8 *.

Definition at line 711 of file Base.h.

13.3.3.2 VA_LIST

```
typedef CHAR8* VA_LIST
```

Variable used to traverse the list of arguments.

This type can vary by implementation and could be an array or structure.

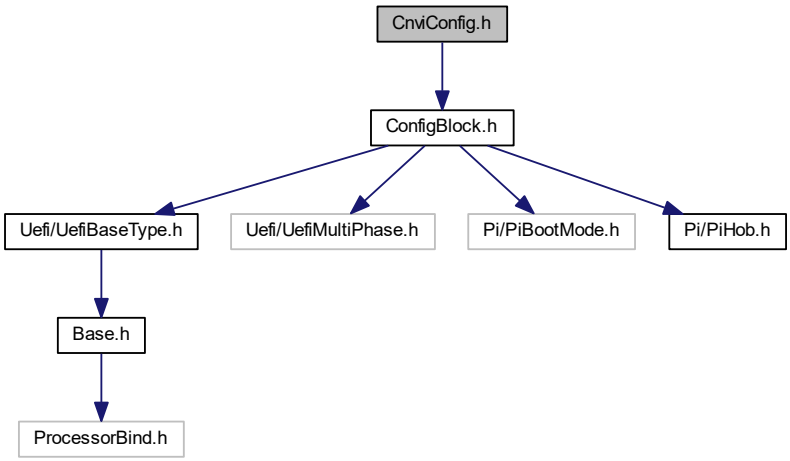
Definition at line 643 of file Base.h.

13.4 CnviConfig.h File Reference

CNVi policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for CnviConfig.h:



Classes

- struct CNVI_PIN_MUX

  CNVi signals pin muxing settings.
- struct CNVI_PREMEM_CONFIG

  The CNVI_PREMEM_CONFIG block describes the expected configuration of the CNVi IP.

Macros

- #define CNVI_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define CNVI_PREMEM_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Enumerations

- enum CNVI_MODE

  CNVi Mode options.
- enum CNVI_BT_INTERFACE

  CNVi Bluetooth interface options.

13.4.1 Detailed Description

CNVi policy.

Copyright

INTEL CONFIDENTIAL Copyright 2016 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

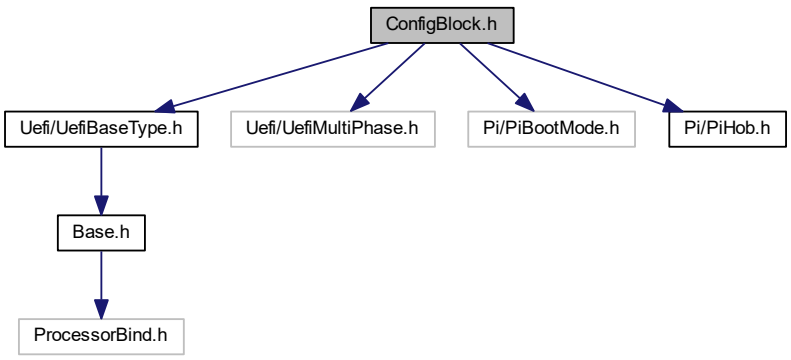
Specification Reference:

13.5 ConfigBlock.h File Reference

Header file for Config Block Lib implementation.

```
#include <Uefi/UefiBaseType.h>
#include <Uefi/UefiMultiPhase.h>
#include <Pi/PiBootMode.h>
#include <Pi/PiHob.h>
```

Include dependency graph for ConfigBlock.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct _CONFIG_BLOCK_HEADER

  Config Block Header.
- struct _CONFIG_BLOCK

  Config Block.
- struct _CONFIG_BLOCK_TABLE_STRUCT

  Config Block Table Header.

Typedefs

- typedef struct _CONFIG_BLOCK_HEADER CONFIG_BLOCK_HEADER

  Config Block Header.
- typedef struct _CONFIG_BLOCK CONFIG_BLOCK

  Config Block.
- typedef struct _CONFIG_BLOCK_TABLE_STRUCT CONFIG_BLOCK_TABLE_HEADER

  Config Block Table Header.

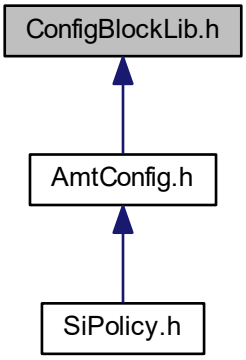
13.5.1 Detailed Description

Header file for Config Block Lib implementation.

13.6 ConfigBlockLib.h File Reference

Header file for Config Block Lib implementation.

This graph shows which files directly or indirectly include this file:



Functions

- EFI_STATUS CreateConfigBlockTable (IN UINT16 TotalSize, OUT VOID **ConfigBlockTableAddress)

  Create config block table.
- EFI_STATUS AddConfigBlock (IN VOID *ConfigBlockTableAddress, OUT VOID **ConfigBlockAddress)

  Add config block into config block table structure.
- EFI_STATUS GetConfigBlock (IN VOID *ConfigBlockTableAddress, IN EFI_GUID *ConfigBlockGuid, OUT VOID **ConfigBlockAddress)

  Retrieve a specific Config Block data by GUID.

13.6.1 Detailed Description

Header file for Config Block Lib implementation.

Copyright

13.6.2 Function Documentation

13.6.2.1 AddConfigBlock()

```
EFI_STATUS AddConfigBlock (
             IN VOID * ConfigBlockTableAddress,
             OUT VOID ** ConfigBlockAddress )
```

Add config block into config block table structure.

Parameters

in	*ConfigBlockTableAddress*	- A pointer to the beginning of Config Block Table Address
out	*ConfigBlockAddress*	- On return, points to a pointer to the beginning of Config Block Address

Return values

EFI_OUT_OF_RESOURCES	- Config Block Table is full and cannot add new Config Block or Config Block Offset Table is full and cannot add new Config Block.
EFI_SUCCESS	- Successfully added Config Block

13.6.2.2 CreateConfigBlockTable()

```
EFI_STATUS CreateConfigBlockTable (
             IN UINT16 TotalSize,
             OUT VOID ** ConfigBlockTableAddress )
```

Create config block table.

Parameters

in	*TotalSize*	- Max size to be allocated for the Config Block Table
out	*ConfigBlockTableAddress*	- On return, points to a pointer to the beginning of Config Block Table Address

Return values

EFI_INVALID_PARAMETER	- Invalid Parameter
EFI_OUT_OF_RESOURCES	- Out of resources
EFI_SUCCESS	- Successfully created Config Block Table at ConfigBlockTableAddress

13.6.2.3 GetConfigBlock()

```
EFI_STATUS GetConfigBlock (
             IN VOID * ConfigBlockTableAddress,
```

```
    IN EFI_GUID * ConfigBlockGuid,
    OUT VOID ** ConfigBlockAddress )
```

Retrieve a specific Config Block data by GUID.

Parameters

in	*ConfigBlockTableAddress*	- A pointer to the beginning of Config Block Table Address
in	*ConfigBlockGuid*	- A pointer to the GUID uses to search specific Config Block
out	*ConfigBlockAddress*	- On return, points to a pointer to the beginning of Config Block Address

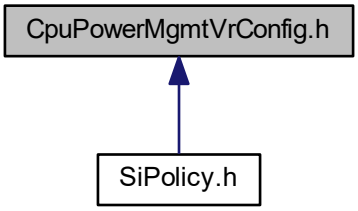
Return values

EFI_NOT_FOUND	- Could not find the Config Block
EFI_SUCCESS	- Config Block found and return

13.7 CpuPowerMgmtVrConfig.h File Reference

CPU Power Management VR Config Block.

This graph shows which files directly or indirectly include this file:



Macros

- #define MAX_NUM_VRS 6

  Defines the maximum number of VR domains supported.
- #define CPU_POWER_MGMT_VR_CONFIG_REVISION 2

  CPU Power Management VR Configuration Structure.

13.7.1 Detailed Description

CPU Power Management VR Config Block.

Specification Reference:

13.7.2 Macro Definition Documentation

13.7.2.1 CPU_POWER_MGMT_VR_CONFIG_REVISION

```
#define CPU_POWER_MGMT_VR_CONFIG_REVISION 2
```

CPU Power Management VR Configuration Structure.

Revision 1:

- Initial version. **Revision 2**:
- Deprecated VccInDemotionEnable, VccInDemotionQuiescentPowerInMw and VccInDemotionCapacitance↩ InUf Added VccDemotionShutdownThreshold

Definition at line 58 of file CpuPowerMgmtVrConfig.h.

13.7.2.2 MAX_NUM_VRS

```
#define MAX_NUM_VRS 6
```

Defines the maximum number of VR domains supported.

Warning

: Changing this define would cause DWORD alignment issues in policy structures.

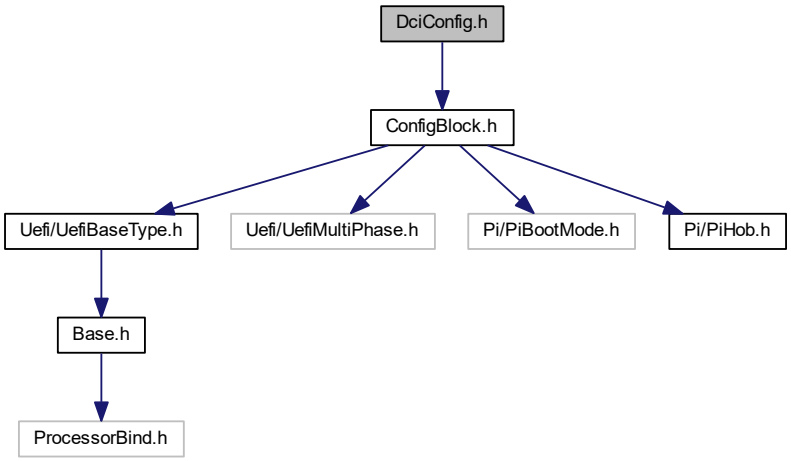
Definition at line 46 of file CpuPowerMgmtVrConfig.h.

13.8 DciConfig.h File Reference

Dci policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for DciConfig.h:



Classes

- struct PCH_DCI_PREMEM_CONFIG

  The PCH_DCI_PREMEM_CONFIG block describes policies related to Direct Connection Interface (DCI)

13.8.1 Detailed Description

Dci policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.9 EspiConfig.h File Reference

Espi policy.

Macros

- #define PCH_ESPI_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.9.1 Detailed Description

Espi policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

Specification Reference:

13.10 FivrConfig.h File Reference

PCH FIVR policy.

Classes

- struct FIVR_EXT_RAIL_CONFIG

  Structure for V1p05/Vnn VR rail configuration.
- struct FIVR_VCCIN_AUX_CONFIG

  Structure for VCCIN_AUX voltage rail configuration.
- struct PCH_FIVR_CONFIG

  The PCH_FIVR_CONFIG block describes FIVR settings.

Enumerations

- enum FIVR_RAIL_SX_STATE

  Rail support in S0ix and Sx Settings other than FivrRailDisabled can be OR'ed.

13.10.1 Detailed Description

PCH FIVR policy.

Copyright

INTEL CONFIDENTIAL Copyright 2017 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.11 FlashProtectionConfig.h File Reference

FlashProtection policy.

Classes

- struct PROTECTED_RANGE

  Protected Flash Range.
- struct PCH_FLASH_PROTECTION_CONFIG

  The PCH provides a method for blocking writes and reads to specific ranges in the SPI flash when the Protected Ranges are enabled.

13.11.1 Detailed Description

FlashProtection policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2020 Intel Corporation.

Specification Reference:

13.12 FspErrorInfo.h File Reference

FSP Error Information HOB to describe errors inside FSP that bootloader may take some actions to handle those error scenarios.

Classes

- struct FSP_ERROR_INFO_HOB

  FSP Error Information Block.

Macros

- #define FSP_ERROR_INFO_HOB_GUID

  GUID value indicating the FSP error information.

13.12.1 Detailed Description

FSP Error Information HOB to describe errors inside FSP that bootloader may take some actions to handle those error scenarios.

Copyright

Specification Reference:

13.13 FspFixedPcds.h File Reference

This file lists all FixedAtBuild PCDs referenced in FSP integration guide.

Macros

- #define PcdFspAreaBaseAddress 0xFFD00000
  FspAreaBaseAddress.
- #define PcdFspImageIdString $MTLFSP$
  FspImageIdString.
- #define PcdSiliconInitVersionMajor 0x0D
  SiliconInitVersionMajor.
- #define PcdSiliconInitVersionMinor 0x00
  SiliconInitVersionMinor.
- #define PcdSiliconInitVersionRevision 0x05
  SiliconInitVersionRevision.
- #define PcdSiliconInitVersionBuild 0x10
  SiliconInitVersionBuild.
- #define PcdGlobalDataPointerAddress 0xFED00148
  GlobalDataPointerAddress.
- #define PcdTemporaryRamBase 0xFEF00000
  TemporaryRamBase.
- #define PcdTemporaryRamSize 0x00100000
  TemporaryRamSize.
- #define PcdFspReservedBufferSize 0x100
  FspReservedBufferSize.

13.13.1 Detailed Description

This file lists all FixedAtBuild PCDs referenced in FSP integration guide.

Those value may vary in different FSP revision to meet different requirements.

13.14 FspmArchConfigPpi.h File Reference

Header file for FSP-M Arch Config PPI.

Classes

- struct FSPM_ARCH_CONFIG_PPI

  This PPI provides FSP-M Arch Config PPI.

Macros

- #define FSPM_ARCH_CONFIG_GUID

  Global ID for the FSPM_ARCH_CONFIG_PPI.

13.14.1 Detailed Description

Header file for FSP-M Arch Config PPI.

13.15 GbeConfig.h File Reference

Gigabit Ethernet policy.

Classes

- struct GBE_CONFIG

  PCH intergrated GBE controller configuration settings.

13.15.1 Detailed Description

Gigabit Ethernet policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2018 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

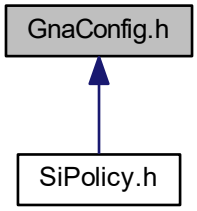
This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.16 GnaConfig.h File Reference

Policy definition for GNA Config Block.

This graph shows which files directly or indirectly include this file:



Classes

- struct GNA_CONFIG

  GNA config block for configuring GNA.

13.16.1 Detailed Description

Policy definition for GNA Config Block.

Copyright

Specification Reference:

13.17 GpioConfig.h File Reference

Header file for GpioConfig structure used by GPIO library.

Classes

- struct GPIO_CONFIG

  GPIO configuration structure used for pin programming.

Macros

- #define B_GPIO_INT_CONFIG_INT_SOURCE_MASK 0x1F

  Mask for GPIO_INT_CONFIG for interrupt source.
- #define B_GPIO_INT_CONFIG_INT_TYPE_MASK 0xE0

  Mask for GPIO_INT_CONFIG for interrupt type.
- #define B_GPIO_ELECTRICAL_CONFIG_TERMINATION_MASK 0x1F

  Mask for GPIO_ELECTRICAL_CONFIG for termination value.
- #define B_GPIO_ELECTRICAL_CONFIG_1V8_TOLERANCE_MASK 0x60

  Mask for GPIO_ELECTRICAL_CONFIG for 1v8 tolerance setting.
- #define B_GPIO_LOCK_CONFIG_PAD_CONF_LOCK_MASK 0x3

  Mask for GPIO_LOCK_CONFIG for Pad Configuration Lock.
- #define B_GPIO_LOCK_CONFIG_OUTPUT_LOCK_MASK 0x5

  Mask for GPIO_LOCK_CONFIG for Pad Output Lock.
- #define B_GPIO_OTHER_CONFIG_RXRAW_MASK 0x3

  Mask for GPIO_OTHER_CONFIG for RxRaw1 setting.

Typedefs

- typedef UINT32 GPIO_PAD

  For any GpioPad usage in code use GPIO_PAD type.
- typedef UINT32 GPIO_GROUP

  For any GpioGroup usage in code use GPIO_GROUP type.

Enumerations

- enum GPIO_HARDWARE_DEFAULT
- enum GPIO_PAD_MODE

  GPIO Pad Mode Refer to GPIO documentation on native functions available for certain pad.
- enum GPIO_HOSTSW_OWN

  Host Software Pad Ownership modes This setting affects GPIO interrupt status registers.
- enum GPIO_DIRECTION

  GPIO Direction.
- enum GPIO_OUTPUT_STATE

  GPIO Output State This field is relevant only if output is enabled.
- enum GPIO_INT_CONFIG

  GPIO interrupt configuration This setting is applicable only if pad is in GPIO mode and has input enabled.
- enum GPIO_RESET_CONFIG

  GPIO Power Configuration GPIO_RESET_CONFIG allows to set GPIO Reset type (PADCFG_DW0.PadRstCfg) which will be used to reset certain GPIO settings.
- enum GPIO_ELECTRICAL_CONFIG

  GPIO Electrical Configuration Set GPIO termination and Pad Tolerance (applicable only for some pads) Field from GpioTermNone to GpioTermNative can be OR'ed with GpioTolerance1v8.
- enum GPIO_LOCK_CONFIG

  GPIO LockConfiguration Set GPIO configuration lock and output state lock.
- enum GPIO_OTHER_CONFIG

  Other GPIO Configuration GPIO_OTHER_CONFIG is used for less often settings and for future extensions Supported settings:

13.17.1 Detailed Description

Header file for GpioConfig structure used by GPIO library.

Copyright

INTEL CONFIDENTIAL Copyright 2014 - 2017 Intel Corporation.

Specification Reference:

13.17.2 Enumeration Type Documentation

13.17.2.1 GPIO_DIRECTION

`enum GPIO_DIRECTION`

GPIO Direction.

Enumerator

GpioDirDefault	Leave pad direction setting unmodified.
GpioDirInOut	Set pad for both output and input.
GpioDirInInvOut	Set pad for both output and input with inversion.
GpioDirIn	Set pad for input only.
GpioDirInInv	Set pad for input with inversion.
GpioDirOut	Set pad for output only.
GpioDirNone	Disable both output and input.

Definition at line 167 of file GpioConfig.h.

13.17.2.2 GPIO_ELECTRICAL_CONFIG

`enum GPIO_ELECTRICAL_CONFIG`

GPIO Electrical Configuration Set GPIO termination and Pad Tolerance (applicable only for some pads) Field from GpioTermNone to GpioTermNative can be OR'ed with GpioTolerance1v8.

Enumerator

GpioTermDefault	Leave termination setting unmodified.
GpioTermNone	none
GpioTermWpd5K	5kOhm weak pull-down
GpioTermWpd20K	20kOhm weak pull-down
GpioTermWpu1K	1kOhm weak pull-up
GpioTermWpu2K	2kOhm weak pull-up
GpioTermWpu5K	5kOhm weak pull-up
GpioTermWpu20K	20kOhm weak pull-up
GpioTermWpu1K2K	1kOhm & 2kOhm weak pull-up
GpioTermNative	Native function controls pads termination This setting is applicable only to some native modes. Please check EDS to determine which native functionality can control pads termination
GpioNoTolerance1v8	Disable 1.8V pad tolerance.
GpioTolerance1v8	Enable 1.8V pad tolerance.

Definition at line 289 of file GpioConfig.h.

13.17.2.3 GPIO_HARDWARE_DEFAULT

enum GPIO_HARDWARE_DEFAULT

Enumerator

GpioHardwareDefault	Leave setting unmodified.

Definition at line 118 of file GpioConfig.h.

13.17.2.4 GPIO_HOSTSW_OWN

enum GPIO_HOSTSW_OWN

Host Software Pad Ownership modes This setting affects GPIO interrupt status registers.

Depending on chosen ownership some GPIO Interrupt status register get updated and other masked. Please refer to EDS for HOSTSW_OWN register description.

Enumerator

GpioHostOwnDefault	Leave ownership value unmodified.
GpioHostOwnAcpi	Set HOST ownership to ACPI. Use this setting if pad is not going to be used by GPIO OS driver. If GPIO is configured to generate SCI/SMI/NMI then this setting must be used for interrupts to work
GpioHostOwnGpio	Set HOST ownership to GPIO Driver mode. Use this setting only if GPIO pad should be controlled by GPIO OS Driver. GPIO OS Driver will be able to control the pad if appropriate entry in ACPI exists (refer to ACPI specification for GpioIo and GpioInt descriptors)

Definition at line 146 of file GpioConfig.h.

13.17.2.5 GPIO_INT_CONFIG

enum GPIO_INT_CONFIG

GPIO interrupt configuration This setting is applicable only if pad is in GPIO mode and has input enabled.

GPIO_INT_CONFIG allows to choose which interrupt is generated (IOxAPIC/SCI/SMI/NMI) and how it is triggered (edge or level). Refer to PADCFG_DW0 register description in EDS for details on this settings. Field from GpioIntNmi to GpioIntApic can be OR'ed with GpioIntLevel to GpioIntBothEdge to describe an interrupt e.g. GpioIntApic

| GpioIntLevel If GPIO is set to cause an SCI then also GPI_GPE_EN is enabled for this pad. If GPIO is set to cause an NMI then also GPI_NMI_EN is enabled for this pad. Not all GPIO are capable of generating an SMI or NMI interrupt. When routing GPIO to cause an IOxAPIC interrupt care must be taken, as this interrupt cannot be shared and its IRQn number is not configurable. Refer to EDS for GPIO pads IRQ numbers (PADCFG_DW1.IntSel) If GPIO is under GPIO OS driver control and appropriate ACPI GpioInt descriptor exist then use only trigger type setting (from GpioIntLevel to GpioIntBothEdge). This type of GPIO Driver interrupt doesn't have any additional routing setting required to be set by BIOS. Interrupt is handled by GPIO OS Driver.

Enumerator

GpioIntDefault	Leave value of interrupt routing unmodified.
GpioIntDis	Disable IOxAPIC/SCI/SMI/NMI interrupt generation.
GpioIntNmi	Enable NMI interrupt only.
GpioIntSmi	Enable SMI interrupt only.
GpioIntSci	Enable SCI interrupt only.
GpioIntApic	Enable IOxAPIC interrupt only.
GpioIntLevel	Set interrupt as level triggered.
GpioIntEdge	Set interrupt as edge triggered (type of edge depends on input inversion)
GpioIntLvlEdgDis	Disable interrupt trigger.
GpioIntBothEdge	Set interrupt as both edge triggered.

Definition at line 207 of file GpioConfig.h.

13.17.2.6 GPIO_LOCK_CONFIG

`enum GPIO_LOCK_CONFIG`

GPIO LockConfiguration Set GPIO configuration lock and output state lock.

GpioLockPadConfig and GpioLockOutputState can be OR'ed. Lock settings reset is in Powergood domain. Care must be taken when using this setting as fields it locks may be reset by a different signal and can be controllable by what is in GPIO_RESET_CONFIG (PADCFG_DW0.PadRstCfg). GPIO library provides functions which allow to unlock a GPIO pad.

Enumerator

GpioLockDefault	Leave lock setting unmodified.
GpioPadConfigLock	Lock Pad Configuration.
GpioOutputStateLock	Lock GPIO pad output value.

Definition at line 322 of file GpioConfig.h.

13.17.2.7 GPIO_OTHER_CONFIG

`enum GPIO_OTHER_CONFIG`

Other GPIO Configuration GPIO_OTHER_CONFIG is used for less often settings and for future extensions Supported settings:

- RX raw override to '1' - allows to override input value to '1' This setting is applicable only if in input mode (both in GPIO and native usage). The override takes place at the internal pad state directly from buffer and before the RXINV.

Enumerator

GpioRxRaw1Default	Use default input override value.
GpioRxRaw1Dis	Don't override input.
GpioRxRaw1En	Override input to '1'.

Definition at line 339 of file GpioConfig.h.

13.17.2.8 GPIO_OUTPUT_STATE

`enum GPIO_OUTPUT_STATE`

GPIO Output State This field is relevant only if output is enabled.

Enumerator

GpioOutDefault	Leave output value unmodified.
GpioOutLow	Set output to low.
GpioOutHigh	Set output to high.

Definition at line 181 of file GpioConfig.h.

13.17.2.9 GPIO_PAD_MODE

`enum GPIO_PAD_MODE`

GPIO Pad Mode Refer to GPIO documentation on native functions available for certain pad.

If GPIO is set to one of NativeX modes then following settings are not applicable and can be skipped:

- Interrupt related settings
- Host Software Ownership
- Output/Input enabling/disabling
- Output lock

Definition at line 132 of file GpioConfig.h.

13.17.2.10 GPIO_RESET_CONFIG

```
enum GPIO_RESET_CONFIG
```

GPIO Power Configuration GPIO_RESET_CONFIG allows to set GPIO Reset type (PADCFG_DW0.PadRstCfg) which will be used to reset certain GPIO settings.

Refer to EDS for settings that are controllable by PadRstCfg.

Enumerator

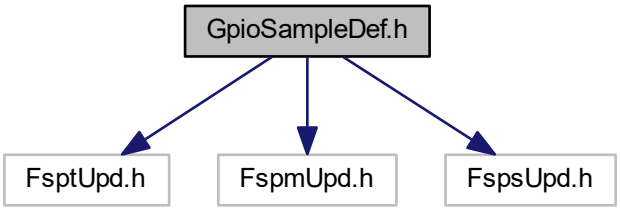
GpioResetDefault	Leave value of pad reset unmodified.
GpioResumeReset	New GPIO reset configuration options. Resume Reset (RSMRST) GPP: PadRstCfg = 00b = "Powergood" GPD: PadRstCfg = 11b = "Resume Reset" Pad setting will reset on: • DeepSx transition • G3 Pad settings will not reset on: • S3/S4/S5 transition • Warm/Cold/Global reset
GpioHostDeepReset	Host Deep Reset PadRstCfg = 01b = "Deep GPIO Reset" Pad settings will reset on: • Warm/Cold/Global reset • DeepSx transition • G3 Pad settings will not reset on: • S3/S4/S5 transition
GpioPlatformReset	Platform Reset (PLTRST) PadRstCfg = 10b = "GPIO Reset" Pad settings will reset on: • S3/S4/S5 transition • Warm/Cold/Global reset • DeepSx transition • G3
GpioDswReset	Deep Sleep Well Reset (DSW_PWROK) GPP: not applicable GPD: PadRstCfg = 00b = "Powergood" Pad settings will reset on: • G3 Pad settings will not reset on: • S3/S4/S5 transition • Warm/Cold/Global reset • DeepSx transition

Definition at line 229 of file GpioConfig.h.

13.18 GpioSampleDef.h File Reference


```
#include <FsptUpd.h>
#include <FspmUpd.h>
```

```
#include <FspsUpd.h>
```
Include dependency graph for GpioSampleDef.h:



13.18.1 Detailed Description

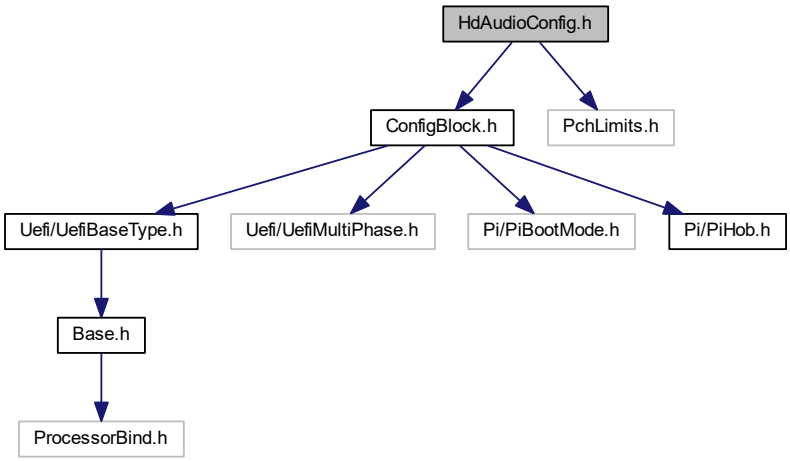
This file is automatically generated. Please do NOT modify !!!

13.19 HdAudioConfig.h File Reference

HDAUDIO policy.

```
#include <ConfigBlock.h>
#include <PchLimits.h>
```

Include dependency graph for HdAudioConfig.h:



Classes

- struct HDA_VERB_TABLE_HEADER

  Azalia verb table header Every verb table should contain this defined header and followed by azalia verb commands.
- struct HDA_LINK_HDA

  HD Audio Link Policies.
- struct HDA_LINK_DMIC

  HD Audio DMIC Interface Policies.
- struct HDA_LINK_SSP

  HD Audio SSP Interface Policies.
- struct HDA_LINK_SNDW

  HD Audio SNDW Interface Policies.
- struct HDAUDIO_PREMEM_CONFIG

  This structure contains the premem policies which are related to HD Audio device (cAVS).
- struct HDAUDIO_DXE_CONFIG

  This structure contains the DXE policies which are related to HD Audio device (cAVS).

Macros

- #define HDAUDIO_PREMEM_CONFIG_REVISION 2

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define HDAUDIO_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define HDAUDIO_DXE_CONFIG_REVISION 1

Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
***Revision 1**: - Initial version.*

- #define HDAUDIO_VERB_TABLE_VIDDID(Vid, Did) (UINT32)((UINT16)Vid | ((UINT16)Did << 16))

  The PCH_HDAUDIO_CONFIG block describes the expected configuration of the Intel HD Audio feature.
- #define HDAUDIO_VERB_TABLE_INIT(Vid, Did, Rid, Sdi, ...)

  Use this macro to create HDAUDIO_VERB_TABLE and populate size automatically.

13.19.1 Detailed Description

HDAUDIO policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2022 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.19.2 Macro Definition Documentation

13.19.2.1 HDAUDIO_PREMEM_CONFIG_REVISION

```
#define HDAUDIO_PREMEM_CONFIG_REVISION 2
```

Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
Revision 1: - Initial version.

Revision 2: - Add HdaDiscBtOffload.

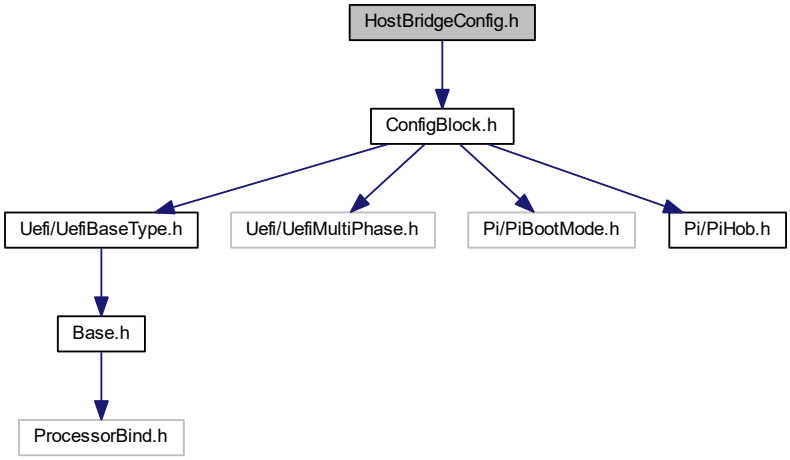
Definition at line 48 of file HdAudioConfig.h.

13.20 HostBridgeConfig.h File Reference

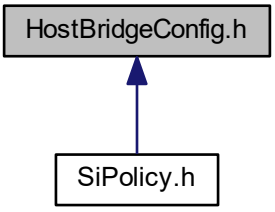
Configurations for HostBridge.

```
#include <ConfigBlock.h>
```

Include dependency graph for HostBridgeConfig.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define HOST_BRIDGE_PREMEM_CONFIG_REVISION 2

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define HOST_BRIDGE_PEI_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.20.1 Detailed Description

Configurations for HostBridge.

Copyright

INTEL CONFIDENTIAL Copyright 2020 - 2021 Intel Corporation.

Specification Reference:

13.20.2 Macro Definition Documentation

13.20.2.1 HOST_BRIDGE_PREMEM_CONFIG_REVISION

```
#define HOST_BRIDGE_PREMEM_CONFIG_REVISION 2
```

Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
Revision 1: - Initial version.

Revision 2: - Added RootBridgeReservedMmio. **Revision 3** - Added RootBridgeReservedIo.

Definition at line 48 of file HostBridgeConfig.h.

13.21 HsioConfig.h File Reference

HSIO policy.

Classes

- struct PCH_HSIO_PREMEM_CONFIG

  The PCH_HSIO_PREMEM_CONFIG block provides HSIO message related settings.
- struct PCH_HSIO_CONFIG

  The PCH_HSIO_CONFIG block provides HSIO message related settings.

Macros

- #define PCH_HSIO_PREMEM_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define PCH_HSIO_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.21.1 Detailed Description

HSIO policy.

Copyright

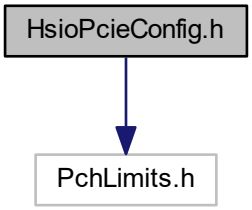
INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

Specification Reference:

13.22 HsioPcieConfig.h File Reference

HSIO pcie policy.

`#include <PchLimits.h>`
Include dependency graph for HsioPcieConfig.h:



Classes

- struct PCH_HSIO_PCIE_LANE_CONFIG

  The PCH_HSIO_PCIE_LANE_CONFIG describes HSIO settings for PCIe lane.
- struct PCH_HSIO_PCIE_PREMEM_CONFIG

  The PCH_HSIO_PCIE_CONFIG block describes the configuration of the HSIO for PCIe lanes.

13.22.1 Detailed Description

HSIO pcie policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2020 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.23 HsioSataConfig.h File Reference

Hsio Sata policy.

Classes

- struct PCH_HSIO_SATA_PORT_LANE

  The PCH_HSIO_SATA_PORT_LANE describes HSIO settings for SATA Port lane.
- struct PCH_HSIO_SATA_PREMEM_CONFIG

  The PCH_HSIO_SATA_CONFIG block describes the HSIO configuration of the SATA controller.

13.23.1 Detailed Description

Hsio Sata policy.

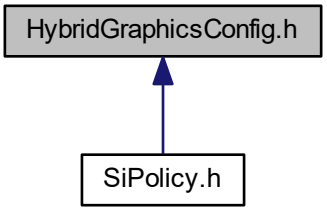
Copyright

Specification Reference:

13.24 HybridGraphicsConfig.h File Reference

Hybrid Graphics policy definitions.

This graph shows which files directly or indirectly include this file:



Macros

- #define HYBRID_GRAPHICS_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.24.1 Detailed Description

Hybrid Graphics policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright 2014 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

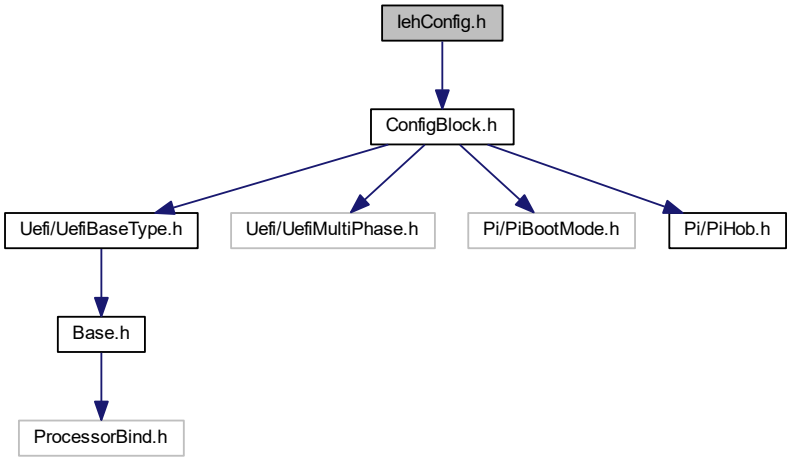
Specification Reference:

13.25 IehConfig.h File Reference

Integrated Error Handler policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for IehConfig.h:



Classes

- struct IEH_CONFIG

  The IEH_CONFIG block describes the expected configuration of the PCH Integrated Error Handler.

13.25.1 Detailed Description

Integrated Error Handler policy.

Copyright

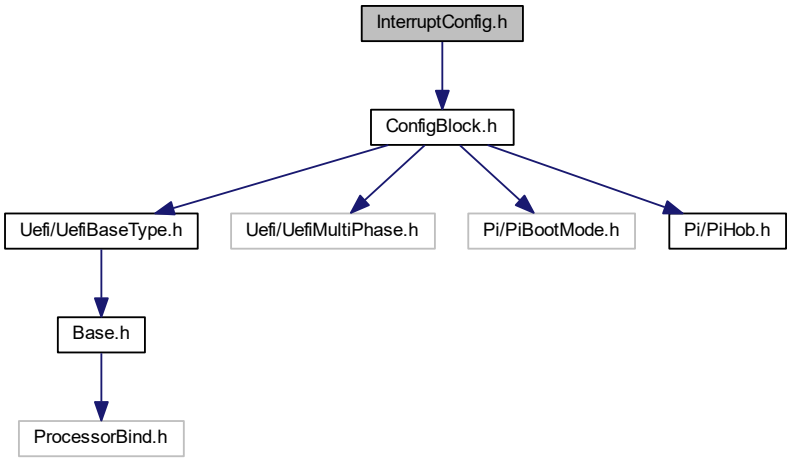
Specification Reference:

13.26 InterruptConfig.h File Reference

Interrupt policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for InterruptConfig.h:



Classes

- struct PCH_DEVICE_INTERRUPT_CONFIG

  The PCH_DEVICE_INTERRUPT_CONFIG block describes interrupt pin, IRQ and interrupt mode for PCH device.
- struct PCH_INTERRUPT_CONFIG

  The PCH_INTERRUPT_CONFIG block describes interrupt settings for PCH.

Macros

- #define PCH_MAX_DEVICE_INTERRUPT_CONFIG 128

  Number of all PCH devices.
- #define PCH_MAX_PXRC_CONFIG 8

  Number of PXRC registers in ITSS.
- #define PCH_MAX_ITSS_IPC_REGS 4

  Number of IPC registers in ITSS.
- #define PCH_MAX_ITSS_IRQ_NUM 120

  Maximum number of IRQs.

Enumerations

- enum PCH_INT_PIN

13.26.1 Detailed Description

Interrupt policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2020 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.26.2 Enumeration Type Documentation

13.26.2.1 PCH_INT_PIN

```
enum PCH_INT_PIN
```

Enumerator

PchNoInt	No Interrupt Pin.

Definition at line 48 of file InterruptConfig.h.

13.27 IoApicConfig.h File Reference

IoApic policy.

Classes

- struct PCH_IOAPIC_CONFIG

  The PCH_IOAPIC_CONFIG block describes the expected configuration of the PCH IO APIC, it's optional and PCH code would ignore it if the BdfValid bit is not TRUE.

13.27.1 Detailed Description

IoApic policy.

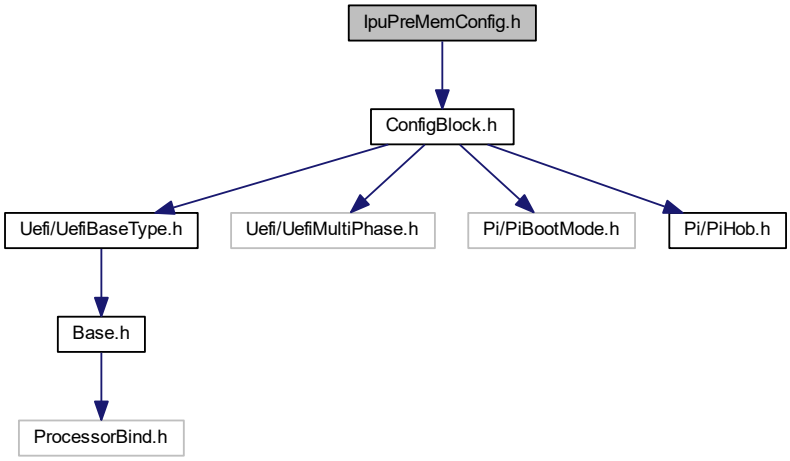
Copyright

Specification Reference:

13.28 IpuPreMemConfig.h File Reference

IPU policy definitions.

`#include <ConfigBlock.h>`
Include dependency graph for IpuPreMemConfig.h:



Classes

- struct IPU_PREMEM_CONFIG
  IPU PreMem configuration
  ***Revision 1**:*

13.28.1 Detailed Description

IPU policy definitions.

Copyright

INTEL CONFIDENTIAL Copyright 2017 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.29 IshConfig.h File Reference

ISH policy.

Classes

- struct ISH_GPIO_CONFIG

  ISH GPIO settings.
- struct ISH_SPI_PIN_CONFIG

  SPI signals settings.
- struct ISH_UART_PIN_CONFIG

  UART signals settings.
- struct ISH_I2C_PIN_CONFIG

  I2C signals settings.
- struct ISH_SPI

  Struct contains GPIO pins assigned and signal settings of SPI.
- struct ISH_UART

  Struct contains GPIO pins assigned and signal settings of UART.
- struct ISH_I2C

  Struct contains GPIO pins assigned and signal settings of I2C.
- struct ISH_GP

  Struct contains GPIO pins assigned and signal settings of GP.
- struct ISH_CONFIG

  The ISH_CONFIG block describes Integrated Sensor Hub device.
- struct ISH_I3C_CONFIG

  The ISH_I3C_CONFIG block describes I3C in ISH device.
- struct ISH_PREMEM_CONFIG

  Premem Policy for Integrated Sensor Hub device.

Macros

- #define ISH_PREMEM_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define ISH_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define ISH_I3C_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.29.1 Detailed Description

ISH policy.

Copyright

Specification Reference:

13.30 LpcConfig.h File Reference

Lpc policy.

Classes

- struct PCH_LPC_PREMEM_CONFIG

  This structure contains the policies which are related to LPC.

13.30.1 Detailed Description

Lpc policy.

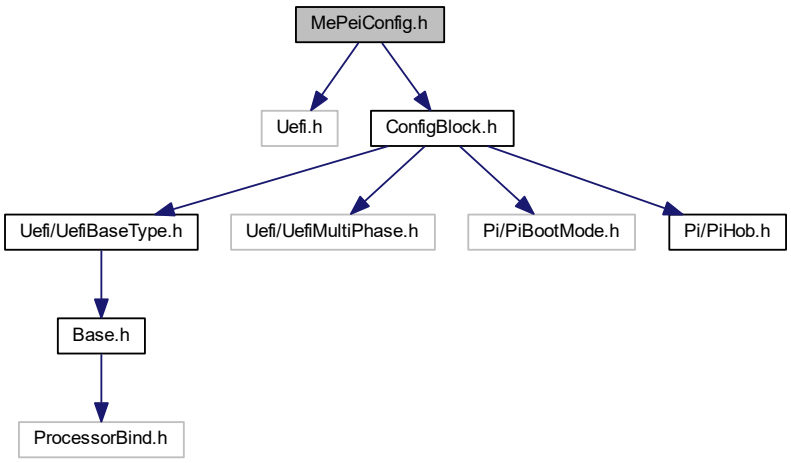
Specification Reference:

13.31 MePeiConfig.h File Reference

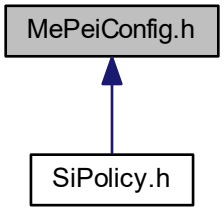
ME config block for PEI phase.

```
#include <Uefi.h>
#include <ConfigBlock.h>
```

Include dependency graph for MePeiConfig.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct ME_PEI_PREMEM_CONFIG

  ME Pei Pre-Memory Configuration Structure.
- struct ME_PEI_CONFIG

  ME Pei Post-Memory Configuration Structure.

13.31.1 Detailed Description

ME config block for PEI phase.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

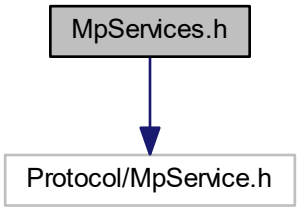
Specification Reference:

13.32 MpServices.h File Reference

This file declares UEFI PI Multi-processor PPI.

```
#include <Protocol/MpService.h>
```

Include dependency graph for MpServices.h:



Classes

- struct _EFI_PEI_MP_SERVICES_PPI

  This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

Typedefs

- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, OUT UINTN ∗NumberOfProcessors, OUT UINTN ∗NumberOfEnabledProcessors)

  Get the number of CPU's.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN UINTN ProcessorNumber, OUT EFI_PROCESSOR_INFORMATION ∗ProcessorInfoBuffer)

  Get information on a specific CPU.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_STARTUP_ALL_APS) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN EFI_AP_PROCEDURE Procedure, IN BOOLEAN SingleThread, IN UINTN TimeoutInMicroSeconds, IN VOID ∗ProcedureArgument OPTIONAL)

  Activate all of the application processors.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_STARTUP_THIS_AP) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN EFI_AP_PROCEDURE Procedure, IN UINTN ProcessorNumber, IN UINTN TimeoutInMicroseconds, IN VOID ∗ProcedureArgument OPTIONAL)

  Activate a specific application processor.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_SWITCH_BSP) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN UINTN ProcessorNumber, IN BOOLEAN EnableOldBSP)

  Switch the boot strap processor.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_ENABLEDISABLEAP) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, IN UINTN ProcessorNumber, IN BOOLEAN EnableAP, IN UINT32 ∗HealthFlag OPTIONAL)

  Enable or disable an application processor.
- typedef EFI_STATUS(∗ EFI_PEI_MP_SERVICES_WHOAMI) (IN CONST EFI_PEI_SERVICES ∗∗PeiServices, IN EFI_PEI_MP_SERVICES_PPI ∗This, OUT UINTN ∗ProcessorNumber)

  Identify the currently executing processor.

13.32.1 Detailed Description

This file declares UEFI PI Multi-processor PPI.

This PPI is installed by some platform or chipset-specific PEIM that abstracts handling multiprocessor support.

Revision Reference:

This PPI is introduced in PI Version 1.4.

13.32.2 Typedef Documentation

13.32.2.1 EFI_PEI_MP_SERVICES_ENABLEDISABLEAP

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_ENABLEDISABLEAP) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN UINTN ProcessorNumber, IN BOOLEAN EnableAP, IN
UINT32 *HealthFlag OPTIONAL)
```

Enable or disable an application processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().
in	*EnableAP*	Specifies the new state for the processor for enabled, FALSE for disabled.
in	*HealthFlag*	If not NULL, a pointer to a value that specifies the new health status of the AP. This flag corresponds to StatusFlag defined in EFI_PEI_MP_SERVICES_PPI.GetProcessorInfo(). Only the PROCESSOR_HEALTH_STATUS_BIT is used. All other bits are ignored. If it is NULL, this parameter is ignored.

Return values

EFI_SUCCESS	The specified AP was enabled or disabled successfully.
EFI_UNSUPPORTED	Enabling or disabling an AP cannot be completed prior to this service returning.
EFI_UNSUPPORTED	Enabling or disabling an AP is not supported.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_NOT_FOUND	Processor with the handle specified by ProcessorNumber does not exist.
EFI_INVALID_PARAMETER	ProcessorNumber specifies the BSP.

Definition at line 229 of file MpServices.h.

13.32.2.2 EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_GET_NUMBER_OF_PROCESSORS) (IN CONST EFI_PEI_SERVI↩
CES **PeiServices, IN EFI_PEI_MP_SERVICES_PPI *This, OUT UINTN *NumberOfProcessors, OUT UINTN
*NumberOfEnabledProcessors)
```

Get the number of CPU's.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	Pointer to this instance of the PPI.
out	*NumberOfProcessors*	Pointer to the total number of logical processors in the system, including the BSP and disabled APs.
out	*NumberOfEnabledProcessors*	Number of processors in the system that are enabled.

Return values

EFI_SUCCESS	The number of logical processors and enabled logical processors was retrieved.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_INVALID_PARAMETER	NumberOfProcessors is NULL. NumberOfEnabledProcessors is NULL.

Definition at line 44 of file MpServices.h.

13.32.2.3 EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_GET_PROCESSOR_INFO) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN UINTN ProcessorNumber, OUT EFI_PROCESSOR_INFOR↩
MATION *ProcessorInfoBuffer)
```

Get information on a specific CPU.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	Pointer to this instance of the PPI.
in	*ProcessorNumber*	Pointer to the total number of logical processors in the system, including the BSP and disabled APs.
out	*ProcessorInfoBuffer*	Number of processors in the system that are enabled.

Return values

EFI_SUCCESS	Processor information was returned.
EFI_DEVICE_ERROR	The calling processor is an AP.

Return values

EFI_INVALID_PARAMETER	ProcessorInfoBuffer is NULL.
EFI_NOT_FOUND	The processor with the handle specified by ProcessorNumber does not exist in the platform.

Definition at line 69 of file MpServices.h.

13.32.2.4 EFI_PEI_MP_SERVICES_STARTUP_ALL_APS

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_STARTUP_ALL_APS) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN EFI_AP_PROCEDURE Procedure, IN BOOLEAN Single↩
Thread, IN UINTN TimeoutInMicroSeconds, IN VOID *ProcedureArgument OPTIONAL)
```

Activate all of the application processors.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*Procedure*	A pointer to the function to be run on enabled APs of the system.
in	*SingleThread*	If TRUE, then all the enabled APs execute the function specified by Procedure one by one, in ascending order of processor handle number. If FALSE, then all the enabled APs execute the function specified by Procedure simultaneously.
in	*TimeoutInMicroSeconds*	Indicates the time limit in microseconds for APs to return from Procedure, for blocking mode only. Zero means infinity. If the timeout expires before all APs return from Procedure, then Procedure on the failed APs is terminated. All enabled APs are available for next function assigned by EFI_PEI_MP_SERVICES_PPI.StartupAllAPs() or EFI_PEI_MP_SERVICES_PPI.StartupThisAP(). If the timeout expires in blocking mode, BSP returns EFI_TIMEOUT.
in	*ProcedureArgument*	The parameter passed into Procedure for all APs.

Return values

EFI_SUCCESS	In blocking mode, all APs have finished before the timeout expired.
EFI_DEVICE_ERROR	Caller processor is AP.
EFI_NOT_STARTED	No enabled APs exist in the system.
EFI_NOT_READY	Any enabled APs are busy.
EFI_TIMEOUT	In blocking mode, the timeout expired before all enabled APs have finished.
EFI_INVALID_PARAMETER	Procedure is NULL.

Definition at line 112 of file MpServices.h.

13.32.2.5 EFI_PEI_MP_SERVICES_STARTUP_THIS_AP

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_STARTUP_THIS_AP) (IN CONST EFI_PEI_SERVICES **Pei↩
Services, IN EFI_PEI_MP_SERVICES_PPI *This, IN EFI_AP_PROCEDURE Procedure, IN UINTN Processor↩
Number, IN UINTN TimeoutInMicroseconds, IN VOID *ProcedureArgument OPTIONAL)
```

Activate a specific application processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*Procedure*	A pointer to the function to be run on enabled APs of the system.
in	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().
in	*TimeoutInMicroSeconds*	Indicates the time limit in microseconds for APs to return from Procedure, for blocking mode only. Zero means infinity. If the timeout expires before all APs return from Procedure, then Procedure on the failed APs is terminated. All enabled APs are available for next function assigned by EFI_PEI_MP_SERVICES_PPI.StartupAllAPs() or EFI_PEI_MP_SERVICES_PPI.StartupThisAP(). If the timeout expires in blocking mode, BSP returns EFI_TIMEOUT.
in	*ProcedureArgument*	The parameter passed into Procedure for all APs.

Return values

EFI_SUCCESS	In blocking mode, specified AP finished before the timeout expires.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_TIMEOUT	In blocking mode, the timeout expired before the specified AP has finished.
EFI_NOT_FOUND	The processor with the handle specified by ProcessorNumber does not exist.
EFI_INVALID_PARAMETER	ProcessorNumber specifies the BSP or disabled AP.
EFI_INVALID_PARAMETER	Procedure is NULL.

Definition at line 157 of file MpServices.h.

13.32.2.6 EFI_PEI_MP_SERVICES_SWITCH_BSP

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_SWITCH_BSP) (IN CONST EFI_PEI_SERVICES **PeiServices,
IN EFI_PEI_MP_SERVICES_PPI *This, IN UINTN ProcessorNumber, IN BOOLEAN EnableOldBSP)
```

Switch the boot strap processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
in	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().
in	*EnableOldBSP*	If TRUE, then the old BSP will be listed as an enabled AP. Otherwise, it will be disabled.

Return values

EFI_SUCCESS	BSP successfully switched.
EFI_UNSUPPORTED	Switching the BSP cannot be completed prior to this service returning.
EFI_UNSUPPORTED	Switching the BSP is not supported.
EFI_DEVICE_ERROR	The calling processor is an AP.
EFI_NOT_FOUND	The processor with the handle specified by ProcessorNumber does not exist.
EFI_INVALID_PARAMETER	ProcessorNumber specifies the current BSP or a disabled AP.
EFI_NOT_READY	The specified AP is busy.

Definition at line 192 of file MpServices.h.

13.32.2.7 EFI_PEI_MP_SERVICES_WHOAMI

```
typedef EFI_STATUS( * EFI_PEI_MP_SERVICES_WHOAMI) (IN CONST EFI_PEI_SERVICES **PeiServices, IN
EFI_PEI_MP_SERVICES_PPI *This, OUT UINTN *ProcessorNumber)
```

Identify the currently executing processor.

Parameters

in	*PeiServices*	An indirect pointer to the PEI Services Table published by the PEI Foundation.
in	*This*	A pointer to the EFI_PEI_MP_SERVICES_PPI instance.
out	*ProcessorNumber*	The handle number of the AP. The range is from 0 to the total number of logical processors minus 1. The total number of logical processors can be retrieved by EFI_PEI_MP_SERVICES_PPI.GetNumberOfProcessors().

Return values

EFI_SUCCESS	The current processor handle number was returned in ProcessorNumber.
EFI_INVALID_PARAMETER	ProcessorNumber is NULL.

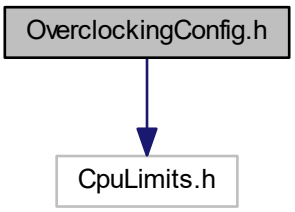
Definition at line 254 of file MpServices.h.

13.33 OverclockingConfig.h File Reference

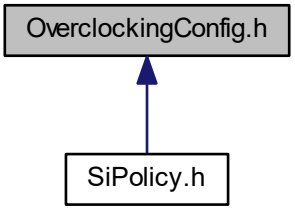
Overclocking Config Block.

```
#include <CpuLimits.h>
```

Include dependency graph for OverclockingConfig.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct OVERCLOCKING_PREMEM_CONFIG

  Overclocking Configuration Structure.

13.33.1 Detailed Description

Overclocking Config Block.

Copyright

INTEL CONFIDENTIAL Copyright 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and

Specification Reference:

13.34 P2sbConfig.h File Reference

P2sb policy.

Classes

- struct PCH_P2SB_CONFIG

  This structure contains the policies which are related to P2SB device.

13.34.1 Detailed Description

P2sb policy.

Copyright

Specification Reference:

13.35 PchDmiConfig.h File Reference

DMI policy.

Classes

- struct PCH_DMI_CONFIG

  The PCH_DMI_CONFIG block describes the expected configuration of the PCH for DMI.

13.35.1 Detailed Description

DMI policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.36 PchGeneralConfig.h File Reference

PCH General policy.

Classes

- struct PCH_GENERAL_CONFIG

  *PCH General Configuration **Revision 2**: - Add VtdEnabled field.*

Macros

- #define PCH_GENERAL_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define PCH_GENERAL_PREMEM_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Enumerations

- enum PCH_RESERVED_PAGE_ROUTE

13.36.1 Detailed Description

PCH General policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.36.2 Enumeration Type Documentation

13.36.2.1 PCH_RESERVED_PAGE_ROUTE

```
enum PCH_RESERVED_PAGE_ROUTE
```

Enumerator

PchReservedPageToLpc	Port 80h cycles are sent to LPC.
PchReservedPageToPcie	Port 80h cycles are sent to PCIe.

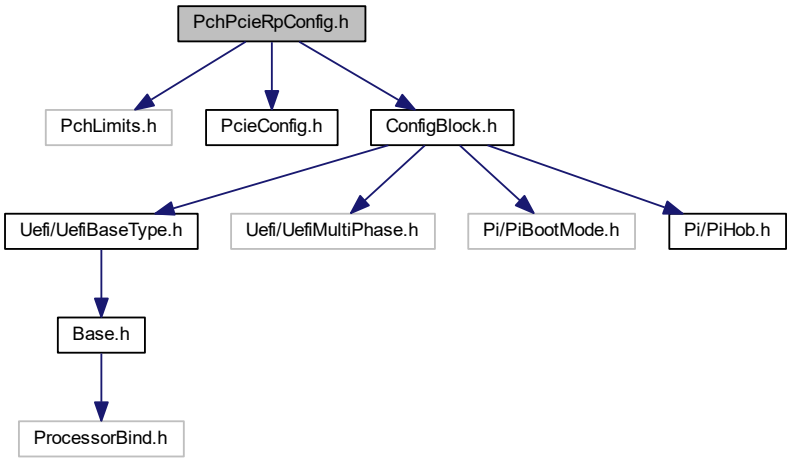
Definition at line 58 of file PchGeneralConfig.h.

13.37 PchPcieRpConfig.h File Reference

PCH Pcie root port policy.

```
#include <PchLimits.h>
#include <PcieConfig.h>
#include <ConfigBlock.h>
```

Include dependency graph for PchPcieRpConfig.h:



Classes

- struct PCH_PCIE_CLOCK

  PCH_PCIE_CLOCK describes PCIe source clock generated by PCH.
- struct DMI_LANE_CONFIG

  The PCH_DMI_LANE_CONFIG block describes specific lane configuration.
- struct PCH_PCIE_ROOT_PORT_CONFIG

  The PCH_PCI_EXPRESS_ROOT_PORT_CONFIG describe the feature and capability of each PCH PCIe root port.
- struct PCH_PCIE_CONFIG

  *The PCH_PCIE_CONFIG block describes the expected configuration of the PCH PCI Express controllers **Revision 1**:*
- struct PCH_PCIE_RP_PREMEM_CONFIG

  *The PCH_PCIE_RP_PREMEM_CONFIG block describes early configuration of the PCH PCI Express controllers **Revision 1**:*
- struct PCIE_RP_DXE_CONFIG

  The PCIE_RP_DXE_CONFIG block describes the expected configuration of the PCH PCI Express controllers in DXE phase.

Macros

- #define PCH_PCIE_CONFIG_REVISION 2

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define PCH_PCIE_RP_PREMEM_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define PCIE_RP_DXE_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Enumerations

- enum PCH_PCIE_ASPM_CONTROL

  The values before AutoConfig match the setting of PCI Express Base Specification 1.1, please be careful for adding new feature.
- enum PCH_PCIE_L1SUBSTATES_CONTROL

  Refer to PCH EDS for the PCH implementation values corresponding to below PCI-E spec defined ranges.
- enum PCH_PCIE_CLOCK_USAGE

13.37.1 Detailed Description

PCH Pcie root port policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

Specification Reference:

13.37.2 Macro Definition Documentation

13.37.2.1 PCH_PCIE_CONFIG_REVISION

```
#define PCH_PCIE_CONFIG_REVISION 2
```

Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
Revision 1: - Initial version.

Revision 2:

- Moved EnablePort8xhDecode policy to PCIE_COMMON_CONFIG
- Add DmiPowerGatingDis

Definition at line 51 of file PchPcieRpConfig.h.

13.37.3 Enumeration Type Documentation

13.37.3.1 PCH_PCIE_CLOCK_USAGE

```
enum PCH_PCIE_CLOCK_USAGE
```

Enumerator

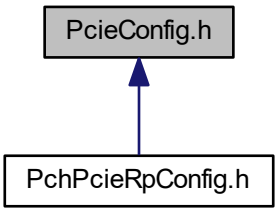
PchClockUsageCpuPcie0	Quantity of PCH and CPU PCIe ports, as well as their encoding in this enum, may change between silicon generations and series. Do not assume that PCH port 0 will be always encoded by 0. Instead, it is recommended to use (PchClockUsagePchPcie0 + PchPortIndex) style to be forward-compatible
PchClockUsageUnspecified	In use for a purpose not listed above.

Definition at line 116 of file PchPcieRpConfig.h.

13.38 PcieConfig.h File Reference

PCIe Config Block.

This graph shows which files directly or indirectly include this file:



Classes

- struct PCIE_EQ_PARAM

  Represent lane specific PCIe Gen3 equalization parameters.
- struct PCIE_LINK_EQ_PLATFORM_SETTINGS

  PCIe Link EQ Platform Settings.
- struct PCIE_COMMON_CONFIG

  PCIe Common Config.
- struct PCIE_DEVICE_OVERRIDE

  PCIe device table entry entry.

Enumerations

- enum PCIE_FORM_FACTOR

  Specifies the form factor that the slot implements.
- enum PCIE_LINK_EQ_METHOD
- enum PCIE_LINK_EQ_MODE

13.38.1 Detailed Description

PCIe Config Block.

Copyright

INTEL CONFIDENTIAL Copyright 2018 - 2021 Intel Corporation.

Specification Reference:

13.38.2 Enumeration Type Documentation

13.38.2.1 PCIE_FORM_FACTOR

`enum PCIE_FORM_FACTOR`

Specifies the form factor that the slot implements.

For custom form factors that do not require any special handling please set PcieFormFactorOther.

Definition at line 102 of file PcieConfig.h.

13.38.2.2 PCIE_LINK_EQ_METHOD

`enum PCIE_LINK_EQ_METHOD`

Enumerator

PcieLinkHardwareEq	Hardware is responsible for performing coefficient/preset search.
PcieLinkFixedEq	No coefficient/preset search is performed. Fixed values are used.

Definition at line 113 of file PcieConfig.h.

13.38.2.3 PCIE_LINK_EQ_MODE

`enum PCIE_LINK_EQ_MODE`

Enumerator

PcieLinkEqPresetMode	Use presets during PCIe link equalization.
PcieLinkEqCoefficientMode	Use coefficients during PCIe link equalization.

Definition at line 118 of file PcieConfig.h.

13.39 PciePreMemConfig.h File Reference

PCIe Config Block PreMem.

Classes

- struct PCIE_IMR_CONFIG

  PCIe IMR Config.
- struct PCIE_PREMEM_CONFIG

  PCIe Pre-Memory Configuration ***Revision 1****: - Initial version.*

13.39.1 Detailed Description

PCIe Config Block PreMem.

Copyright

INTEL CONFIDENTIAL Copyright 2018 - 2020 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

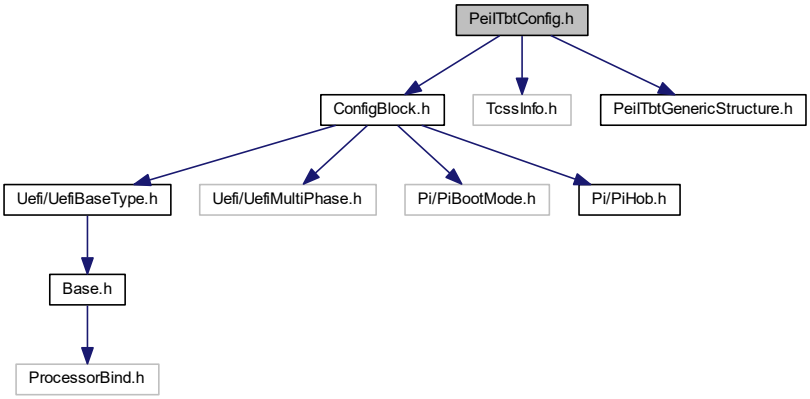
Specification Reference:

13.40 PeiITbtConfig.h File Reference

Header file for TBT PEI Policy.

```
#include <ConfigBlock.h>
#include <TcssInfo.h>
#include <PeiITbtGenericStructure.h>
```

Include dependency graph for PeiITbtConfig.h:



Macros

- #define PEI_ITBT_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.40.1 Detailed Description

Header file for TBT PEI Policy.

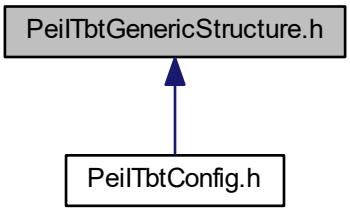
Copyright

Specification Reference:

13.41 PeiITbtGenericStructure.h File Reference

ITBT Policy definition to be referred in both PEI and DXE phase.

This graph shows which files directly or indirectly include this file:



Classes

- struct _ITBT_ROOTPORT_CONFIG

  iTBT RootPort Data Structure
- struct _ITBT_GENERIC_CONFIG

  ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

Typedefs

- typedef struct _ITBT_ROOTPORT_CONFIG ITBT_ROOTPORT_CONFIG

  iTBT RootPort Data Structure
- typedef struct _ITBT_GENERIC_CONFIG ITBT_GENERIC_CONFIG

  ITBT Controller Data Structure to be used cross to RP and controller to be shared by CONFIG_BLOCK and HOB.

13.41.1 Detailed Description

ITBT Policy definition to be referred in both PEI and DXE phase.

Specification Reference:

13.42 PeiPreMemSiDefaultPolicy.h File Reference

This file defines the function to initialize default silicon policy PPI.

Classes

- struct _PEI_PREMEM_SI_DEFAULT_POLICY_INIT_PPI

  This PPI provides function to install default silicon policy.

Typedefs

- typedef EFI_STATUS(∗ PEI_PREMEM_POLICY_INIT) (VOID)

  Initialize and install default silicon policy PPI.

13.42.1 Detailed Description

This file defines the function to initialize default silicon policy PPI.

Copyright

INTEL CONFIDENTIAL Copyright 2019 - 2020 Intel Corporation.

Specification Reference:

13.43 PeiSiDefaultPolicy.h File Reference

This file defines the function to initialize default silicon policy PPI.

Classes

- struct _PEI_SI_DEFAULT_POLICY_INIT_PPI

  This PPI provides function to install default silicon policy.

Typedefs

- typedef EFI_STATUS(∗ PEI_POLICY_INIT) (VOID)

  Initialize and install default silicon policy PPI.

13.43.1 Detailed Description

This file defines the function to initialize default silicon policy PPI.

Copyright

INTEL CONFIDENTIAL Copyright 2019 - 2020 Intel Corporation.

Specification Reference:

13.44 PiHob.h File Reference

HOB related definitions in PI.

This graph shows which files directly or indirectly include this file:

Classes

- struct EFI_HOB_GENERIC_HEADER

  Describes the format and size of the data inside the HOB.
- struct EFI_HOB_HANDOFF_INFO_TABLE

  Contains general state information used by the HOB producer phase.
- struct EFI_HOB_MEMORY_ALLOCATION_HEADER

  EFI_HOB_MEMORY_ALLOCATION_HEADER describes the various attributes of the logical memory allocation.
- struct EFI_HOB_MEMORY_ALLOCATION

  Describes all memory ranges used during the HOB producer phase that exist outside the HOB list.

- struct EFI_HOB_MEMORY_ALLOCATION_STACK

  Describes the memory stack that is produced by the HOB producer phase and upon which all post-memory-installed executable content in the HOB producer phase is executing.
- struct EFI_HOB_MEMORY_ALLOCATION_BSP_STORE

  Defines the location of the boot-strap processor (BSP) BSPStore ("Backing Store Pointer Store").
- struct EFI_HOB_MEMORY_ALLOCATION_MODULE

  Defines the location and entry point of the HOB consumer phase.
- struct EFI_HOB_RESOURCE_DESCRIPTOR

  Describes the resource properties of all fixed, nonrelocatable resource ranges found on the processor host bus during the HOB producer phase.
- struct EFI_HOB_GUID_TYPE

  Allows writers of executable content in the HOB producer phase to maintain and manage HOBs with specific GUID.
- struct EFI_HOB_FIRMWARE_VOLUME

  Details the location of firmware volumes that contain firmware files.
- struct EFI_HOB_FIRMWARE_VOLUME2

  Details the location of a firmware volume that was extracted from a file within another firmware volume.
- struct EFI_HOB_FIRMWARE_VOLUME3

  Details the location of a firmware volume that was extracted from a file within another firmware volume.
- struct EFI_HOB_CPU

  Describes processor information, such as address space and I/O space capabilities.
- struct EFI_HOB_MEMORY_POOL

  Describes pool memory allocations.
- struct EFI_HOB_UEFI_CAPSULE

  Each UEFI capsule HOB details the location of a UEFI capsule.
- union EFI_PEI_HOB_POINTERS

  Union of all the possible HOB Types.

Macros

- #define EFI_HOB_HANDOFF_TABLE_VERSION 0x0009

  Value of version in EFI_HOB_HANDOFF_INFO_TABLE.

Typedefs

- typedef UINT32 EFI_RESOURCE_TYPE

  The resource type.
- typedef UINT32 EFI_RESOURCE_ATTRIBUTE_TYPE

  A type of recount attribute type.

13.44.1 Detailed Description

HOB related definitions in PI.

SPDX-License-Identifier: BSD-2-Clause-Patent

Revision Reference:

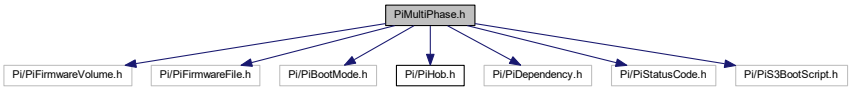
PI Version 1.6

13.45 PiMultiPhase.h File Reference

Include file matches things in PI for multiple module types.

```
#include <Pi/PiFirmwareVolume.h>
#include <Pi/PiFirmwareFile.h>
#include <Pi/PiBootMode.h>
#include <Pi/PiHob.h>
#include <Pi/PiDependency.h>
#include <Pi/PiStatusCode.h>
#include <Pi/PiS3BootScript.h>
```

Include dependency graph for PiMultiPhase.h:



Classes

- struct EFI_MMRAM_DESCRIPTOR

  Structure describing a MMRAM region and its accessibility attributes.
- struct _EFI_MM_RESERVED_MMRAM_REGION

  Structure describing a MMRAM region which cannot be used for the MMRAM heap.

Macros

- #define DXE_ERROR(StatusCode) (MAX_BIT | (MAX_BIT >> 2) | StatusCode)

  Produces an error code in the range reserved for use by the Platform Initialization Architecture Specification.
- #define EFI_REQUEST_UNLOAD_IMAGE DXE_ERROR (1)

  If this value is returned by an EFI image, then the image should be unloaded.
- #define EFI_NOT_AVAILABLE_YET DXE_ERROR (2)

  If this value is returned by an API, it means the capability is not yet installed/available/ready to use.
- #define PI_ENCODE_WARNING(a) ((MAX_BIT >> 2) | (a))

  Success and warning codes reserved for use by PI.
- #define PI_ENCODE_ERROR(a) (MAX_BIT | (MAX_BIT >> 2) | (a))

  Error codes reserved for use by PI.
- #define EFI_INTERRUPT_PENDING PI_ENCODE_ERROR (0)

  Return status codes defined in SMM CIS.
- #define EFI_MMRAM_OPEN 0x00000001

  MMRAM states and capabilities.

- #define EFI_AUTH_STATUS_PLATFORM_OVERRIDE 0x01

  Bitmask of values for Authentication Status.

Typedefs

- typedef struct _EFI_MM_RESERVED_MMRAM_REGION EFI_MM_RESERVED_MMRAM_REGION

  Structure describing a MMRAM region which cannot be used for the MMRAM heap.
- typedef VOID(∗ EFI_AP_PROCEDURE) (IN OUT VOID ∗Buffer)

  The function prototype for invoking a function on an Application Processor.
- typedef EFI_STATUS(∗ EFI_AP_PROCEDURE2) (IN VOID ∗ProcedureArgument)

  The function prototype for invoking a function on an Application Processor.

13.45.1 Detailed Description

Include file matches things in PI for multiple module types.

Revision Reference:

These elements are defined in UEFI Platform Initialization Specification 1.2.

13.45.2 Macro Definition Documentation

13.45.2.1 DXE_ERROR

```
#define DXE_ERROR(
              StatusCode ) (MAX_BIT | (MAX_BIT >> 2) | StatusCode)
```

Produces an error code in the range reserved for use by the Platform Initialization Architecture Specification.

The supported 32-bit range is 0xA0000000-0xBFFFFFFF The supported 64-bit range is 0xA000000000000000-0xBFFFFFFFFFFFFFFF

Parameters

StatusCode	The status code value to convert into a warning code. StatusCode must be in the range 0x00000000..0x1FFFFFFF.

Returns

The value specified by StatusCode in the PI reserved range.

Definition at line 36 of file PiMultiPhase.h.

13.45.2.2 EFI_AUTH_STATUS_PLATFORM_OVERRIDE

```
#define EFI_AUTH_STATUS_PLATFORM_OVERRIDE 0x01
```

Bitmask of values for Authentication Status.

Authentication Status is returned from EFI_GUIDED_SECTION_EXTRACTION_PROTOCOL and the EFI_PEI_↩ GUIDED_SECTION_EXTRACTION_PPI

xx00 Image was not signed. xxx1 Platform security policy override. Assumes the same meaning as 0010 (the image was signed, the signature was tested, and the signature passed authentication test). 0010 Image was signed, the signature was tested, and the signature passed authentication test. 0110 Image was signed and the signature was not tested. 1010 Image was signed, the signature was tested, and the signature failed the authentication test.

Definition at line 84 of file PiMultiPhase.h.

13.45.2.3 PI_ENCODE_ERROR

```
#define PI_ENCODE_ERROR(
              a ) (MAX_BIT | (MAX_BIT >> 2) | (a))
```

Error codes reserved for use by PI.

Supported 32-bit range is 0xa0000000-0xbfffffff. Supported 64-bit range is 0xa000000000000000-0xbfffffffffffffff.

Definition at line 61 of file PiMultiPhase.h.

13.45.2.4 PI_ENCODE_WARNING

```
#define PI_ENCODE_WARNING(
              a ) ((MAX_BIT >> 2) | (a))
```

Success and warning codes reserved for use by PI.

Supported 32-bit range is 0x20000000-0x3fffffff. Supported 64-bit range is 0x2000000000000000-0x3fffffffffffffff.

Definition at line 54 of file PiMultiPhase.h.

13.45.3 Typedef Documentation

13.45.3.1 EFI_AP_PROCEDURE

```
typedef VOID( * EFI_AP_PROCEDURE) (IN OUT VOID *Buffer)
```

The function prototype for invoking a function on an Application Processor.

This definition is used by the UEFI MP Serices Protocol, and the PI SMM System Table.

Parameters

in,out	*Buffer*	The pointer to private data buffer.

Definition at line 190 of file PiMultiPhase.h.

13.45.3.2 EFI_AP_PROCEDURE2

```
typedef EFI_STATUS( * EFI_AP_PROCEDURE2) (IN VOID *ProcedureArgument)
```

The function prototype for invoking a function on an Application Processor.

This definition is used by the UEFI MM MP Serices Protocol.

Parameters

in	*ProcedureArgument*	The pointer to private data buffer.

Return values

EFI_SUCCESS	Excutive the procedure successfully

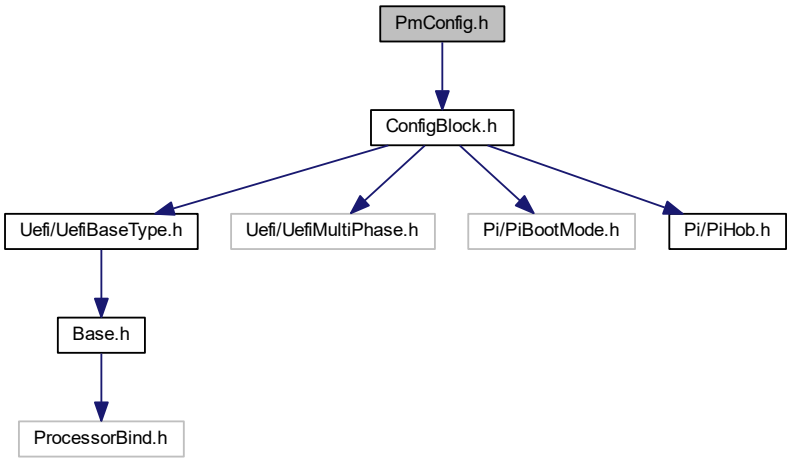
Definition at line 206 of file PiMultiPhase.h.

13.46 PmConfig.h File Reference

Power Management policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for PmConfig.h:



Classes

- struct PCH_WAKE_CONFIG

  This structure allows to customize PCH wake up capability from S5 or DeepSx by WOL, LAN, PCIE wake events.
- union PMC_LPM_S0IX_SUB_STATE_EN

  Low Power Mode Enable config.
- union PMC_GLOBAL_RESET_MASK

  Description of Global Reset Trigger/Event Mask register.
- struct PCH_PM_CONFIG

  The PCH_PM_CONFIG block describes expected miscellaneous power management settings.
- struct BCLK_CONFIG

  BCLK configuration.

Macros

- #define PCH_PM_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define BCLK_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Enumerations

- enum PCH_SLP_S4_MIN_ASSERT

13.46.1 Detailed Description

Power Management policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2021 Intel Corporation.

Specification Reference:

13.46.2 Enumeration Type Documentation

13.46.2.1 PCH_SLP_S4_MIN_ASSERT

```
enum PCH_SLP_S4_MIN_ASSERT
```

Enumerator

PchSlpS4PchTime	The time defined in PCH EDS Power Sequencing and Reset Signal Timings table.

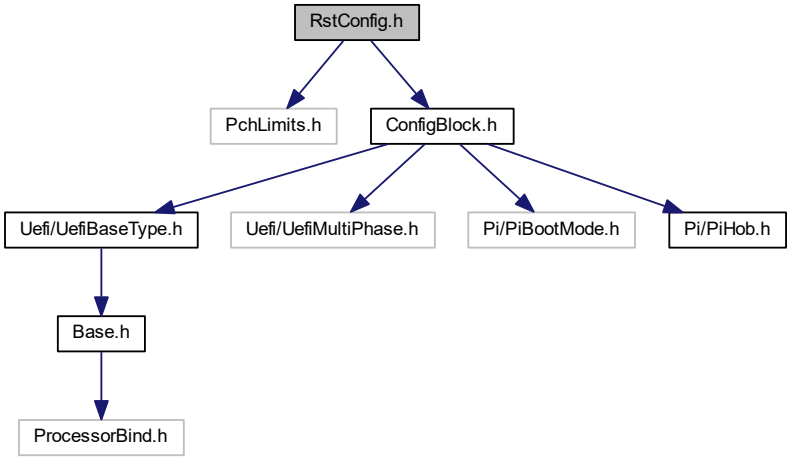
Definition at line 94 of file PmConfig.h.

13.47 RstConfig.h File Reference

Rst policy.

```
#include <PchLimits.h>
#include <ConfigBlock.h>
```

Include dependency graph for RstConfig.h:



Classes

- struct RST_HARDWARE_REMAPPED_STORAGE_CONFIG

  This structure describes the details of Intel RST for PCIe Storage remapping Note: In order to use this feature, Intel RST Driver is required.
- struct RST_CONFIG

  Rapid Storage Technology settings.

13.47.1 Detailed Description

Rst policy.

Copyright

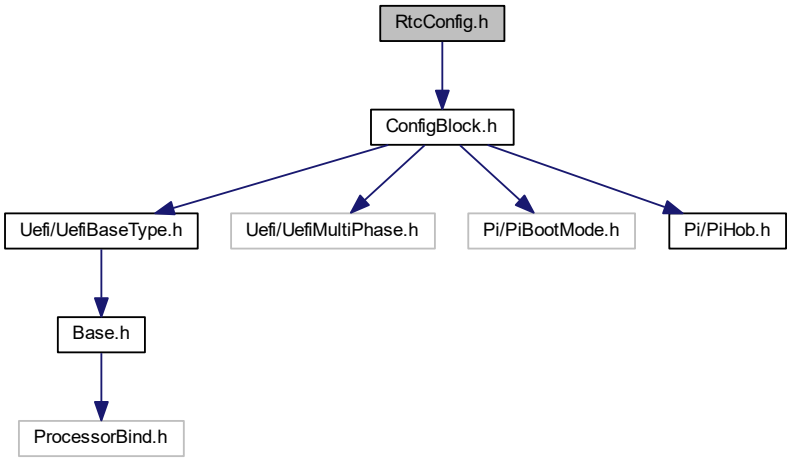
INTEL CONFIDENTIAL Copyright 2018 Intel Corporation.

Specification Reference:

13.48 RtcConfig.h File Reference

RTC policy.

`#include <ConfigBlock.h>`
Include dependency graph for RtcConfig.h:



Classes

- struct RTC_CONFIG

  The RTC_CONFIG block describes the expected configuration of RTC configuration.

13.48.1 Detailed Description

RTC policy.

Copyright

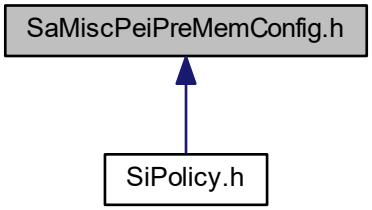
INTEL CONFIDENTIAL Copyright 2018 Intel Corporation.

Specification Reference:

13.49 SaMiscPeiPreMemConfig.h File Reference

Policy details for miscellaneous configuration in System Agent.

This graph shows which files directly or indirectly include this file:



Classes

- struct SA_MISC_PEI_PREMEM_CONFIG

  This configuration block is to configure SA Miscellaneous variables during PEI Pre-Mem phase like programming different System Agent BARs, TsegSize, MmioSize required etc.

13.49.1 Detailed Description

Policy details for miscellaneous configuration in System Agent.

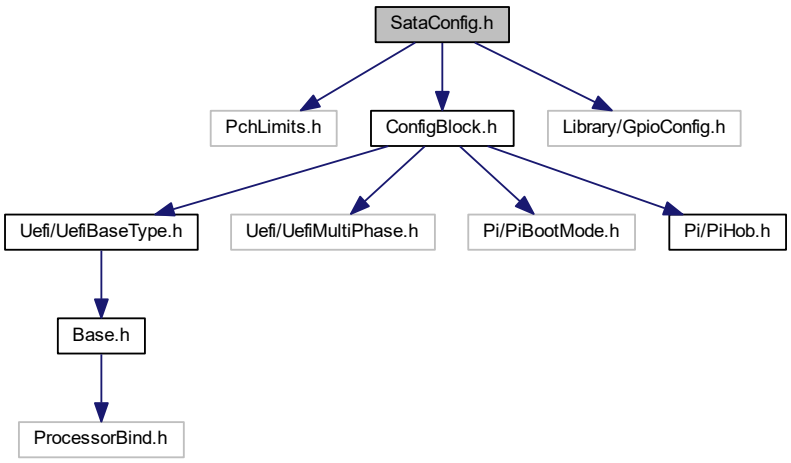
Specification Reference:

13.50 SataConfig.h File Reference

Sata policy.

```
#include <PchLimits.h>
#include <ConfigBlock.h>
#include <Library/GpioConfig.h>
```

Include dependency graph for SataConfig.h:



Classes

- struct PCH_SATA_PORT_CONFIG

  This structure configures the features, property, and capability for each SATA port.
- struct SATA_THERMAL_THROTTLING

  This structure lists PCH supported SATA thermal throttling register setting for customization.
- struct SATA_CONFIG

  The SATA_CONFIG block describes the expected configuration of the SATA controllers.

13.50.1 Detailed Description

Sata policy.

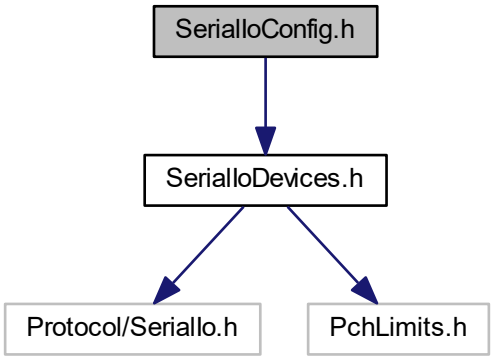
Copyright

Specification Reference:

13.51 SerialIoConfig.h File Reference

Serial IO policy.

```
#include <SerialIoDevices.h>
```
Include dependency graph for SerialIoConfig.h:



Classes

- struct SERIAL_IO_CONFIG

  The SERIAL_IO_CONFIG block provides the configurations to set the Serial IO controllers.

Macros

- #define SERIAL_IO_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.51.1 Detailed Description

Serial IO policy.

Copyright

INTEL CONFIDENTIAL Copyright 2016 - 2022 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

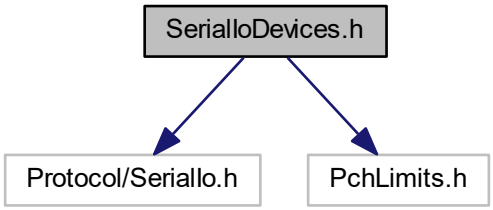
Specification Reference:

13.52 SerialIoDevices.h File Reference

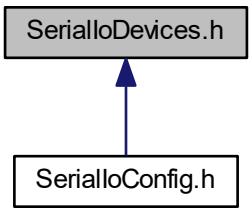
Serial IO policy.

```
#include <Protocol/SerialIo.h>
#include <PchLimits.h>
```

Include dependency graph for SerialIoDevices.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct SPI_PIN_MUX

  SPI signals pin muxing settings.
- struct SERIAL_IO_SPI_CONFIG

  The SERIAL_IO_SPI_CONFIG provides the configurations to set the Serial IO SPI controller.
- struct SERIAL_IO_UART_ATTRIBUTES

  UART Settings.
- struct UART_PIN_MUX

UART signals pin muxing settings.

- struct SERIAL_IO_UART_CONFIG

  Serial IO UART Controller Configuration.

- struct I2C_PIN_MUX

  I2C signals pin muxing settings.

- struct SERIAL_IO_I2C_CONFIG

  Serial IO I2C Controller Configuration.

- struct MUX_GPIO_CONFIG

  I3C GPIO settings.

- struct SERIAL_IO_I3C_CONFIG

  The SERIAL_IO_I3C_CONFIG provides the configurations for Serial IO I3C controller.

Enumerations

- enum SERIAL_IO_SPI_MODE

  Available working modes for SerialIo SPI Host Controller.

- enum SERIAL_IO_CS_POLARITY

  Used to set Inactive/Idle polarity of Spi Chip Select.

- enum SERIAL_IO_UART_MODE

  Available working modes for SerialIo UART Host Controller.

- enum SERIAL_IO_UART_PG
- enum SERIAL_IO_I2C_MODE

  Available working modes for SerialIo I2C Host Controller.

- enum SERIAL_IO_I3C_MODE

  Available working modes for SerialIo I3C Host Controller.

13.52.1 Detailed Description

Serial IO policy.

Copyright

INTEL CONFIDENTIAL Copyright 2016 - 2022 Intel Corporation.

Specification Reference:

13.52.2 Enumeration Type Documentation

13.52.2.1 SERIAL_IO_I2C_MODE

```
enum SERIAL_IO_I2C_MODE
```

Available working modes for SerialIo I2C Host Controller.

```
 0: SerialIoI2cDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: SerialIoI2cPci;

- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled

Definition at line 202 of file SerialIoDevices.h.

13.52.2.2 SERIAL_IO_I3C_MODE

```
enum SERIAL_IO_I3C_MODE
```

Available working modes for SerialIo I3C Host Controller.

```
 0: SerialIoI3cDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - Device will be disabled in PSF
```

1: SerialIoI3cPci;

- Gpio pin configuration in native mode for each assigned pin SDA and SCL
- Device will be enabled in PSF
- Only Bar 0 will be enabled 2: SerialIoI3cPhantom;
- Dedicated mode for I3C1, controls if GPIOs are enabled

Definition at line 252 of file SerialIoDevices.h.

13.52.2.3 SERIAL_IO_SPI_MODE

```
enum SERIAL_IO_SPI_MODE
```

Available working modes for SerialIo SPI Host Controller.

```
 0: SerialIoSpiDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: SerialIoSpiPci;

- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled

Definition at line 58 of file SerialIoDevices.h.

13.52.2.4 SERIAL_IO_UART_MODE

```
enum SERIAL_IO_UART_MODE
```

Available working modes for SerialIo UART Host Controller.

```
 0: SerialIoUartDisabled;
    - Device is placed in D3
    - Gpio configuration is skipped
    - PSF:
    !important! If given device is Function 0 and other higher functions on given device
                are enabled, PSF disabling is skipped. PSF default will remain and device PCI CFG Space will s
                This is needed to allow PCI enumerator access functions above 0 in a multifunction device.
```

1: SerialIoUartPci;

- Designated for Serial IO UART OS driver usage
- Gpio pin configuration in native mode for each assigned pin
- Device will be enabled in PSF
- Only BAR0 will be enabled 2: SerialIoUartHidden;
- Designated for BIOS and/or DBG2 OS kernel debug
- Gpio pin configuration in native mode for each assigned pin
- Device disabled in the PSF
- Both BARs are enabled, BAR1 becomes devices Pci Config Space
- BAR0 assigned from the global PCH reserved memory range, reported as motherboard resource by SIRC

Note

If this controller is located at function 0 and it's mode is set to hidden it will not be visible in the PCI space. 3: SerialIoUartCom;

- Designated for 16550/PNP0501 compatible COM device
- Gpio pin configuration in native mode for each assigned pin
- Device disabled in the PSF
- Both BARs are enabled, BAR1 becomes devices Pci Config Space
- BAR0 assigned from the global PCH reserved memory range, reported as motherboard resource by SIRC 4: SerialIoUartSkipInit;
- Gpio configuration is skipped
- PSF configuration is skipped
- BAR assignemnt is skipped
- D-satate setting is skipped

Definition at line 132 of file SerialIoDevices.h.

13.52.2.5 SERIAL_IO_UART_PG

`enum SERIAL_IO_UART_PG`

Enumerator

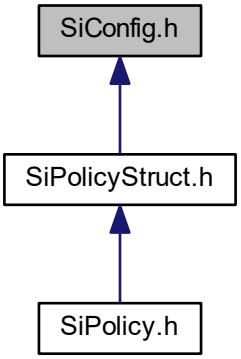
SerialIoUartPgDisabled	No _PS0/_PS3 support, device left in D0, after initialization.
SerialIoUartPgEnabled	In mode: SerialIoUartCom; _PS0/_PS3 that supports getting device out of reset In mode: SerialIoUartPci Keeps UART in D0 and assigns Fixed MMIO for SEC/PEI usage only.
SerialIoUartPgAuto	_PS0 and _PS3, detection through ACPI if device was initialized prior to first PG. If it was used (DBG2) PG is disabled,

Definition at line 175 of file SerialIoDevices.h.

13.53 SiConfig.h File Reference

Si Config Block.

This graph shows which files directly or indirectly include this file:



Classes

- struct SI_CONFIG

  The Silicon Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.
- struct SVID_SID_VALUE

  Subsystem Vendor ID / Subsystem ID.

13.53.1 Detailed Description

Si Config Block.

Copyright

INTEL CONFIDENTIAL Copyright 2016 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.54 SiPolicy.h File Reference

Silicon Policy PPI is used for specifying platform related Intel silicon information and policy setting.

```
#include <SiPolicyStruct.h>
#include <PchPolicyCommon.h>
#include <PchPreMemPolicyCommon.h>
#include <MePeiConfig.h>
#include <AmtConfig.h>
#include <FusaConfig.h>
#include <Uefi.h>
#include <CpuPowerMgmt.h>
#include <CpuInitConfig.h>
#include <CpuPowerMgmtBasicConfig.h>
#include <CpuPowerMgmtCustomConfig.h>
#include <CpuPowerDeliveryConfig.h>
#include <CpuPowerMgmtTestConfig.h>
#include <CpuTestConfig.h>
#include <CpuInitPreMemConfig.h>
#include <CpuSecurityPreMemConfig.h>
#include <TxtConfig.h>
#include <BiosGuardConfig.h>
#include <ConfigBlock/VoltageRegulator/CpuPowerMgmtVrConfig.h>
#include <OverclockingConfig.h>
#include <FiaConfig.h>
#include <Defines/HostBridgeDefines.h>
#include <GraphicsConfig.h>
#include <VtdConfig.h>
#include <GnaConfig.h>
#include <HybridGraphicsConfig.h>
#include <MemoryConfig.h>
#include <ConfigBlock/SaMiscPeiPreMemConfig.h>
#include <TraceHubConfig.h>
#include <HostBridgeConfig.h>
#include <VpuConfig.h>
#include <PeiDmiConfig.h>
#include <DmiConfig.h>
#include <GpioV2PeiConfig.h>
```

Include dependency graph for SiPolicy.h:

13.54.1 Detailed Description

Silicon Policy PPI is used for specifying platform related Intel silicon information and policy setting.

This PPI is consumed by the silicon PEI modules and carried over to silicon DXE modules.

Copyright

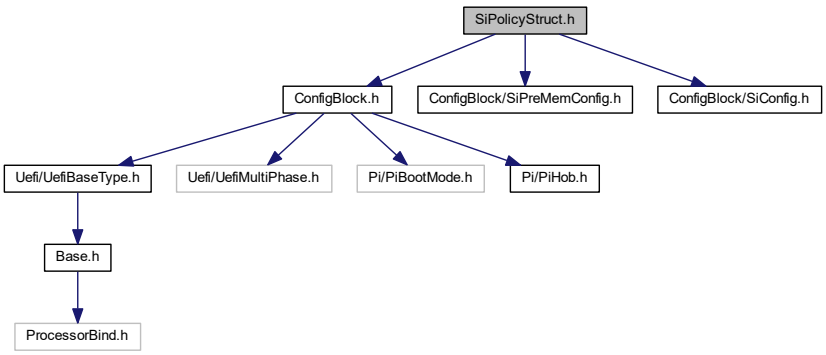
Specification Reference:

13.55 SiPolicyStruct.h File Reference

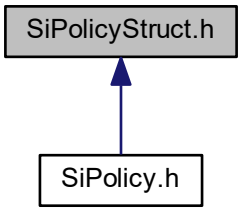
Intel reference code configuration policies.

```
#include <ConfigBlock.h>
#include <ConfigBlock/SiPreMemConfig.h>
#include <ConfigBlock/SiConfig.h>
```

Include dependency graph for SiPolicyStruct.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _SI_PREMEM_POLICY_STRUCT

  SI Policy PPI in Pre-Mem
  All SI config block change history will be listed here

  .
- struct _SI_POLICY_STRUCT

  SI Policy PPI
  All SI config block change history will be listed here

  .

Macros

- #define SI_POLICY_REVISION 1

  Silicon Policy revision number Any change to this structure will result in an update in the revision number.
- #define SI_PREMEM_POLICY_REVISION 1

  Silicon pre-memory Policy revision number Any change to this structure will result in an update in the revision number.

13.55.1 Detailed Description

Intel reference code configuration policies.

Copyright

INTEL CONFIDENTIAL Copyright 2014 - 2020 Intel Corporation.

Specification Reference:

13.55.2 Macro Definition Documentation

13.55.2.1 SI_POLICY_REVISION

```
#define SI_POLICY_REVISION 1
```

Silicon Policy revision number Any change to this structure will result in an update in the revision number.

This member specifies the revision of the Silicon Policy. This field is used to indicate change to the policy structure.

Revision 1:

- Initial version.

Definition at line 52 of file SiPolicyStruct.h.

13.55.2.2 SI_PREMEM_POLICY_REVISION

```
#define SI_PREMEM_POLICY_REVISION 1
```

Silicon pre-memory Policy revision number Any change to this structure will result in an update in the revision number.

Revision 1:

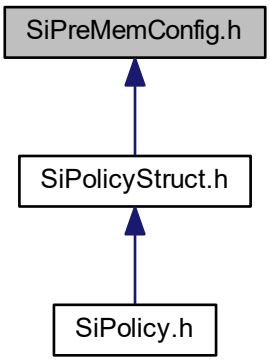
- Initial version.

Definition at line 61 of file SiPolicyStruct.h.

13.56 SiPreMemConfig.h File Reference

Si Config Block PreMem.

This graph shows which files directly or indirectly include this file:



Classes

- struct SI_PREMEM_CONFIG

  The Silicon PreMem Policy allows the platform code to publish a set of configuration information that the RC drivers will use to configure the silicon hardware.

13.56.1 Detailed Description

Si Config Block PreMem.

Copyright

Specification Reference:

13.57 SmbusConfig.h File Reference

Smbus policy.

Classes

- struct PCH_SMBUS_PREMEM_CONFIG

  The SMBUS_CONFIG block lists the reserved addresses for non-ARP capable devices in the platform.

13.57.1 Detailed Description

Smbus policy.

Copyright

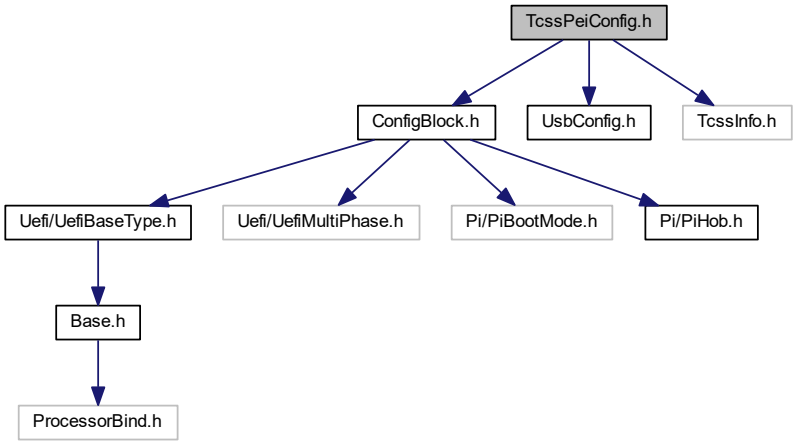
Specification Reference:

13.58 TcssPeiConfig.h File Reference

TCSS PEI policy.

```
#include <ConfigBlock.h>
#include <UsbConfig.h>
#include <TcssInfo.h>
```

Include dependency graph for TcssPeiConfig.h:



Classes

- struct IOM_AUX_ORI_PAD_CONFIG

  The IOM_AUX_ORI_PAD_CONFIG describes IOM TypeC port map GPIO pin.
- struct IOM_INTERFACE_CONFIG

  The IOM_EC_INTERFACE_CONFIG block describes interaction between BIOS and IOM-EC.
- struct PMC_INTERFACE_CONFIG

  The PMC_INTERFACE_CONFIG block describes interaction between BIOS and PMC.
- struct SA_XDCI_IRQ_INT_CONFIG

  The SA XDCI INT Pin and IRQ number.
- struct TCSS_PCIE_PORT_POLICY

  The TCSS_PCIE_PORT_POLICY block describes PCIe settings for TCSS.
- struct TCSS_PCIE_PEI_POLICY

  TCSS_PCIE_PEI_POLICY describes PCIe port settings for TCSS.
- struct TCSS_IOM_ORI_OVERRIDE

  The TCSS_IOM_PEI_CONFIG block describes IOM Aux/HSL override settings for TCSS.
- struct TCSS_IOM_PEI_CONFIG

  The TCSS_IOM_PEI_CONFIG block describes IOM settings for TCSS.
- struct TCSS_MISC_PEI_CONFIG

  The TCSS_MISC_PEI_CONFIG block describes MISC settings for TCSS.
- struct TCSS_PEI_CONFIG

  The TCSS_PEI_CONFIG block describes TCSS settings for SA.

Macros

- #define TCSS_PEI_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.58.1 Detailed Description

TCSS PEI policy.

Copyright

INTEL CONFIDENTIAL Copyright 2016 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

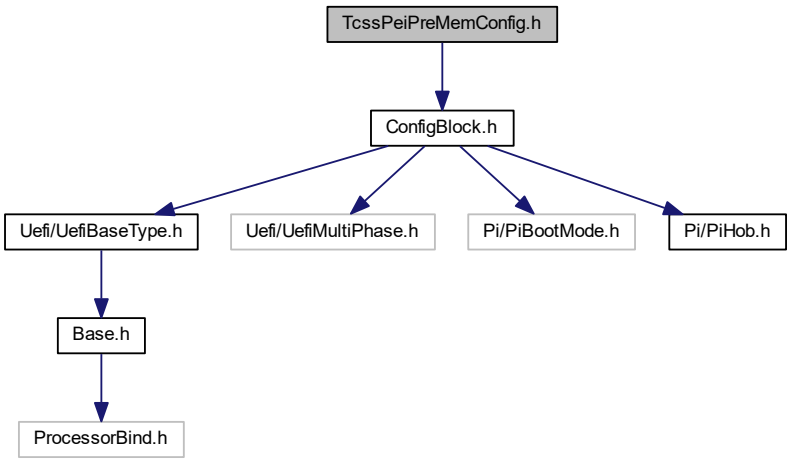
Specification Reference:

13.59 TcssPeiPreMemConfig.h File Reference

TCSS PEI PreMem policy.

```
#include <ConfigBlock.h>
```

Include dependency graph for TcssPeiPreMemConfig.h:



Classes

- union TCSS_DEVEN_PEI_PREMEM_CONFIG

  The TCSS_DEVEN_PEI_PREMEM_CONFIG block describes Device Enable settings for TCSS.
- struct TCSS_USBTC_PEI_PERMEM_CONFIG

  The TCSS_USBTC_PEI_PERMEM_CONFIG block describes IOM settings for TCSS.
- struct TCSS_PEI_PREMEM_CONFIG

  This configuration block describes TCSS settings.

13.59.1 Detailed Description

TCSS PEI PreMem policy.

Copyright

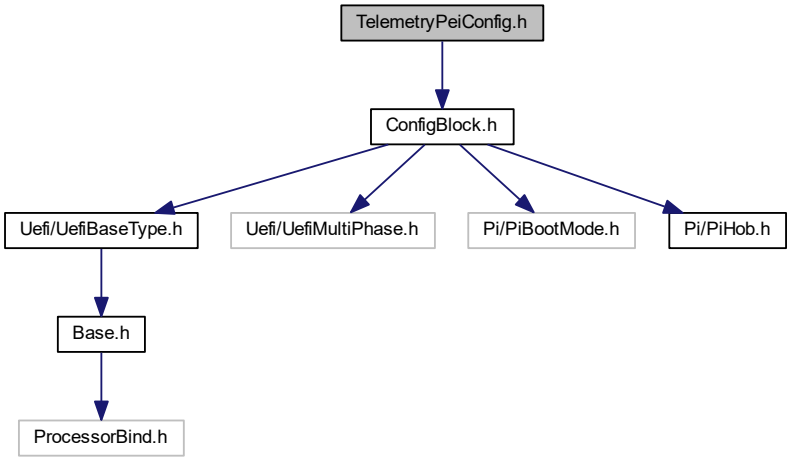
Specification Reference:

13.60 TelemetryPeiConfig.h File Reference

Configurations for Telemetry.

```
#include <ConfigBlock.h>
```

Include dependency graph for TelemetryPeiConfig.h:



Classes

- struct TELEMETRY_PEI_PREMEM_CONFIG

  This configuration block describes Telemetry settings in PreMem.
- struct TELEMETRY_PEI_CONFIG

  This configuration block describes Telemetry settings in PostMem.

13.60.1 Detailed Description

Configurations for Telemetry.

Specification Reference:

13.61 ThcConfig.h File Reference

Touch Host Controller policy.

Classes

- struct THC_HID_OVER_SPI

  THC Hid Over SPI mode related settings.
- struct THC_RESET

  THC Reset Pad related settings.
- struct THC_HID_OVER_I2C

  THC Hid Over I2C mode related settings.
- struct THC_PORT

  Port Configuration structure required for each Port that THC might use.
- struct THC_CONFIG

  THC_CONFIG block provides the configurations forTouch Host Controllers.

Macros

- #define THC_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Enumerations

- enum THC_PORT_ASSIGNMENT

  Available Port Assignments.
- enum THC_MODE

  Available modes.

13.61.1 Detailed Description

Touch Host Controller policy.

Copyright

INTEL CONFIDENTIAL Copyright 2017 - 2022 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.61.2 Enumeration Type Documentation

13.61.2.1 THC_MODE

`enum THC_MODE`

Available modes.

Enumerator

Thc	Intel THC protocol.
HidOverSpi	Hid Over SPI protocol.
HidOverI2c	Hid Over I2C protocol.

Definition at line 63 of file ThcConfig.h.

13.61.2.2 THC_PORT_ASSIGNMENT

```
enum THC_PORT_ASSIGNMENT
```

Available Port Assignments.

Enumerator

ThcAssignmentNone	None of the avaialbe controllers assigned.
ThcAssignmentThc0	Port assigned to THC0.
ThcAssignmentThc1	Port assigned to THC1.

Definition at line 54 of file ThcConfig.h.

13.62 ThermalConfig.h File Reference

Thermal policy.

Classes

- struct THERMAL_THROTTLE_LEVELS

  This structure lists PCH supported throttling register setting for custimization.
- struct DMI_HW_WIDTH_CONTROL

  This structure allows to customize DMI HW Autonomous Width Control for Thermal and Mechanical spec design.
- struct THERMAL_CONFIG

  The THERMAL_CONFIG block describes the expected configuration of the Thermal IP block.

13.62.1 Detailed Description

Thermal policy.

Copyright

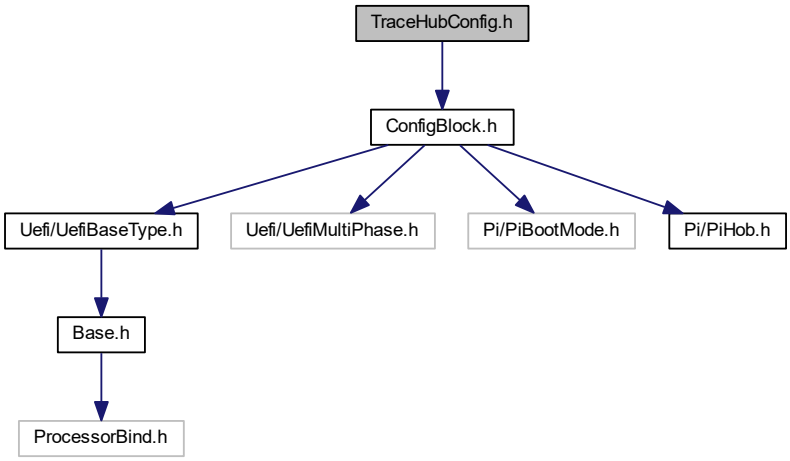
Specification Reference:

13.63 TraceHubConfig.h File Reference

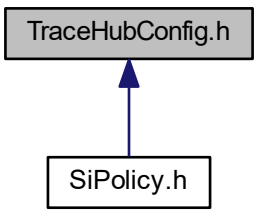
Configurations for TraceHub.

```
#include <ConfigBlock.h>
```

Include dependency graph for TraceHubConfig.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct TRACE_HUB_CONFIG

  TRACE_HUB_CONFIG block describes TraceHub settings.
- struct PCH_TRACE_HUB_PREMEM_CONFIG

  PCH Trace Hub PreMem Configuration Contains Trace Hub settings for PCH side tracing ***Revision 1****: - Initial version.*
- struct TRACE_HUB_PREMEM_CONFIG

  Trace Hub PreMem Configuration Contains Trace Hub settings ***Revision 1****: - Initial version.*

Enumerations

- enum TRACE_HUB_ENABLE_MODE

  The TRACE_HUB_ENABLE_MODE describes TraceHub mode of operation.
- enum TRACE_BUFFER_SIZE

  The TRACE_BUFFER_SIZE describes the desired TraceHub buffer size.

13.63.1 Detailed Description

Configurations for TraceHub.

Copyright

INTEL CONFIDENTIAL Copyright 2018 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

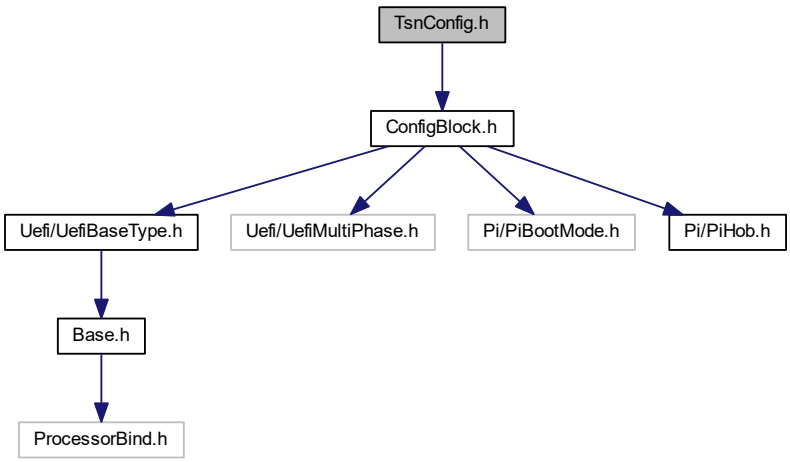
This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

13.64 TsnConfig.h File Reference

TSN Config policy.

`#include <ConfigBlock.h>`
Include dependency graph for TsnConfig.h:



Macros

- #define TSN_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.64.1 Detailed Description

TSN Config policy.

Copyright

INTEL CONFIDENTIAL Copyright 2019 - 2022 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

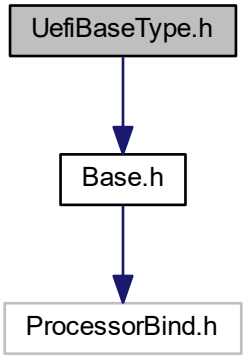
Specification Reference:

13.65 UefiBaseType.h File Reference

Defines data types and constants introduced in UEFI.

```
#include <Base.h>
```

Include dependency graph for UefiBaseType.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct EFI_TIME

  EFI Time Abstraction: Year: 1900 - 9999 Month: 1 - 12 Day: 1 - 31 Hour: 0 - 23 Minute: 0 - 59 Second: 0 - 59 Nanosecond: 0 - 999,999,999 TimeZone: -1440 to 1440 or 2047.

- struct EFI_MAC_ADDRESS

  32-byte buffer containing a network Media Access Control address.

- union EFI_IP_ADDRESS

  16-byte buffer aligned on a 4-byte boundary.

Macros

- #define EFIERR(_a) ENCODE_ERROR(_a)

  Define macro to encode the status code.

- #define EFI_SIZE_TO_PAGES(Size) (((Size) >> EFI_PAGE_SHIFT) + (((Size) & EFI_PAGE_MASK) ? 1 : 0))

  Macro that converts a size, in bytes, to a number of EFI_PAGESs.

- #define EFI_PAGES_TO_SIZE(Pages) ((Pages) << EFI_PAGE_SHIFT)

  Macro that converts a number of EFI_PAGEs to a size in bytes.

- #define EFI_IMAGE_MACHINE_IA32 0x014C

  PE32+ Machine type for IA32 UEFI images.

- #define EFI_IMAGE_MACHINE_IA64 0x0200

  PE32+ Machine type for IA64 UEFI images.

- #define EFI_IMAGE_MACHINE_EBC 0x0EBC

  PE32+ Machine type for EBC UEFI images.

- #define EFI_IMAGE_MACHINE_X64 0x8664

  PE32+ Machine type for X64 UEFI images.

- #define EFI_IMAGE_MACHINE_ARMTHUMB_MIXED 0x01C2

  PE32+ Machine type for ARM mixed ARM and Thumb/Thumb2 images.

- #define EFI_IMAGE_MACHINE_AARCH64 0xAA64

  PE32+ Machine type for AARCH64 A64 images.

- #define EFI_IMAGE_MACHINE_RISCV32 0x5032

  PE32+ Machine type for RISC-V 32/64/128.

- #define EFI_IMAGE_MACHINE_LOONGARCH32 0x6232

  PE32+ Machine type for LoongArch 32/64 images.

- #define EFI_SUCCESS RETURN_SUCCESS

  Enumeration of EFI_STATUS.

- #define EFI_NETWORK_UNREACHABLE EFIERR(100)

  ICMP error definitions.

- #define EFI_CONNECTION_FIN EFIERR(104)

  Tcp connection status definitions.

Typedefs

- typedef GUID EFI_GUID
  128-bit buffer containing a unique identifier value.
- typedef RETURN_STATUS EFI_STATUS
  Function return status for EFI API.
- typedef VOID * EFI_HANDLE
  A collection of related interfaces.
- typedef VOID * EFI_EVENT
  Handle to an event structure.
- typedef UINTN EFI_TPL
  Task priority level.
- typedef UINT64 EFI_LBA
  Logical block address.
- typedef UINT64 EFI_PHYSICAL_ADDRESS
  64-bit physical memory address.
- typedef UINT64 EFI_VIRTUAL_ADDRESS
  64-bit virtual memory address.
- typedef IPv4_ADDRESS EFI_IPv4_ADDRESS
  4-byte buffer.
- typedef IPv6_ADDRESS EFI_IPv6_ADDRESS
  16-byte buffer.

13.65.1 Detailed Description

Defines data types and constants introduced in UEFI.

13.65.2 Macro Definition Documentation

13.65.2.1 EFI_PAGES_TO_SIZE

```
#define EFI_PAGES_TO_SIZE(
              Pages ) ((Pages) << EFI_PAGE_SHIFT)
```

Macro that converts a number of EFI_PAGEs to a size in bytes.

Parameters

Pages	The number of EFI_PAGES. This parameter is assumed to be type UINTN. Passing in a parameter that is larger than UINTN may produce unexpected results.

Returns

The number of bytes associated with the number of EFI_PAGEs specified by Pages.

Definition at line 211 of file UefiBaseType.h.

13.65.2.2 EFI_SIZE_TO_PAGES

```
#define EFI_SIZE_TO_PAGES(
              Size ) (((Size) >> EFI_PAGE_SHIFT) + (((Size) & EFI_PAGE_MASK) ? 1 :  0))
```

Macro that converts a size, in bytes, to a number of EFI_PAGESs.

Parameters

Size	A size in bytes. This parameter is assumed to be type UINTN. Passing in a parameter that is larger than UINTN may produce unexpected results.

Returns

The number of EFI_PAGESs associated with the number of bytes specified by Size.

Definition at line 198 of file UefiBaseType.h.

13.65.3 Typedef Documentation

13.65.3.1 EFI_IPv4_ADDRESS

```
typedef IPv4_ADDRESS EFI_IPv4_ADDRESS
```

4-byte buffer.

An IPv4 internet protocol address.

Definition at line 85 of file UefiBaseType.h.

13.65.3.2 EFI_IPv6_ADDRESS

```
typedef IPv6_ADDRESS EFI_IPv6_ADDRESS
```

16-byte buffer.

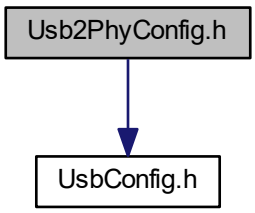
An IPv6 internet protocol address.

Definition at line 90 of file UefiBaseType.h.

13.66 Usb2PhyConfig.h File Reference

USB2 PHY configuration policy.

```
#include <UsbConfig.h>
```
Include dependency graph for Usb2PhyConfig.h:



Classes

- struct USB2_PHY_PARAMETERS

  This structure configures per USB2 AFE settings.
- struct USB2_PHY_CONFIG

  This structure holds info on how to tune electrical parameters of USB2 ports based on board layout.

13.66.1 Detailed Description

USB2 PHY configuration policy.

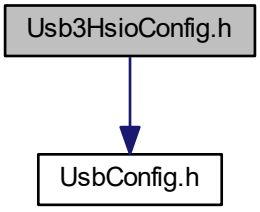
Specification Reference:

13.67 Usb3HsioConfig.h File Reference

USB3 Mod PHY configuration policy.

```
#include <UsbConfig.h>
```

Include dependency graph for Usb3HsioConfig.h:



Classes

- struct HSIO_PARAMETERS

  This structure describes USB3 Port N configuration parameters.

Macros

- #define USB3_HSIO_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

13.67.1 Detailed Description

USB3 Mod PHY configuration policy.

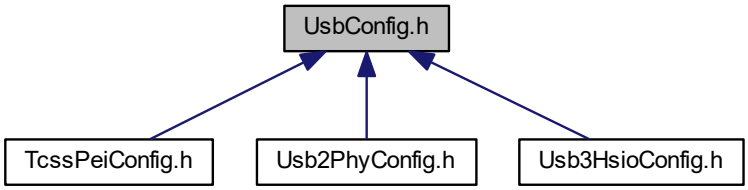
Copyright

Specification Reference:

13.68 UsbConfig.h File Reference

Common USB policy shared between PCH and CPU Contains general features settings for xHCI and xDCI.

This graph shows which files directly or indirectly include this file:



Classes

- struct USB2_PORT_CONFIG

  This structure configures per USB2.0 port settings like enabling and overcurrent protection.
- struct USB3_PORT_CONFIG

  This structure configures per USB3.x port settings like enabling and overcurrent protection.
- struct XDCI_CONFIG

  The XDCI_CONFIG block describes the configurations of the xDCI Usb Device controller.

Macros

- #define USB_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define USB_OC_MAX_PINS 16

  Total OC pins number (both physical and virtual)

13.68.1 Detailed Description

Common USB policy shared between PCH and CPU Contains general features settings for xHCI and xDCI.

Copyright

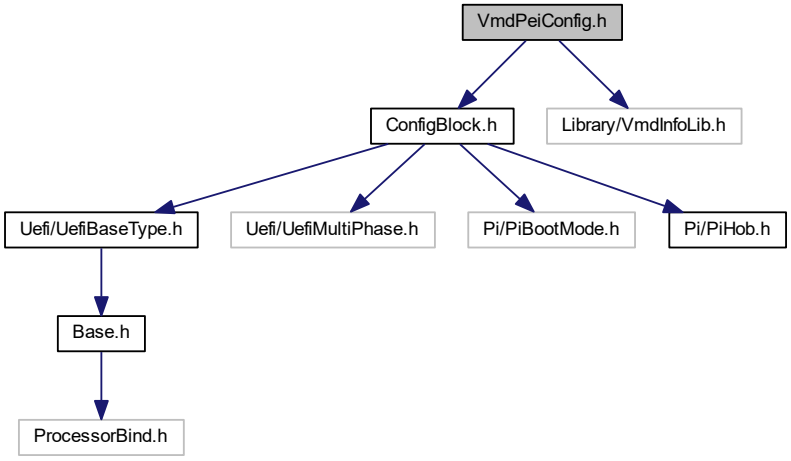
Specification Reference:

13.69 VmdPeiConfig.h File Reference

VMD PEI policy VMD Config Block.

```
#include <ConfigBlock.h>
#include <Library/VmdInfoLib.h>
```

Include dependency graph for VmdPeiConfig.h:



Classes

- struct VMD_PEI_CONFIG

  This configuration block is to configure VMD related variables used in PostMem PEI.

13.69.1 Detailed Description

VMD PEI policy VMD Config Block.

Copyright

INTEL CONFIDENTIAL Copyright 2017 - 2021 Intel Corporation.

The source code contained or described herein and all documents related to the source code ("Material") are owned by Intel Corporation or its suppliers or licensors. Title to the Material remains with Intel Corporation or its suppliers and licensors. The Material may contain trade secrets and proprietary and confidential information of Intel Corporation and its suppliers and licensors, and is protected by worldwide copyright and trade secret laws and treaty provisions. No part of the Material may be used, copied, reproduced, modified, published, uploaded, posted, transmitted, distributed, or disclosed in any way without Intel's prior express written permission.

No license under any patent, copyright, trade secret or other intellectual property right is granted to or conferred upon you by disclosure or delivery of the Materials, either expressly, by implication, inducement, estoppel or otherwise. Any license under such intellectual property rights must be express and approved by Intel in writing.

Unless otherwise agreed by Intel in writing, you may not remove or alter this notice or any other notice embedded in Materials by Intel or Intel's suppliers or licensors in any way.

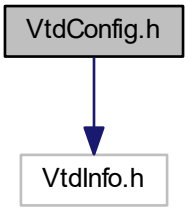
This file contains an 'Intel Peripheral Driver' and is uniquely identified as "Intel Reference Module" and is licensed for Intel CPUs and chipsets under the terms of your license agreement with Intel or your vendor. This file may be modified by the user, subject to additional terms of the license agreement.

Specification Reference:

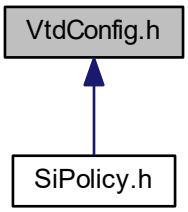
13.70 VtdConfig.h File Reference

VT-d policy definitions.

`#include <VtdInfo.h>`
Include dependency graph for VtdConfig.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct VTD_ENGINE_CONFIG

  This structure describes each VT-d engine.

Macros

- #define VTD_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*
- #define VTD_DXE_CONFIG_REVISION 1

  Making any structure change after API Lock will need to maintain backward compatibility, bump up structure revision and update below history table
  ***Revision 1**: - Initial version.*

Typedefs

- typedef EFI_STATUS(∗ VTD_ENABLE_DMA_BUFFER) (IN VTD_ENGINE VtdEngineNumber)
  Enable DMA buffer in VTd.

13.70.1 Detailed Description

VT-d policy definitions.

Copyright

Specification Reference:

13.70.2 Typedef Documentation

13.70.2.1 VTD_ENABLE_DMA_BUFFER

```
typedef EFI_STATUS( * VTD_ENABLE_DMA_BUFFER) (IN VTD_ENGINE VtdEngineNumber)
```

Enable DMA buffer in VTd.

Parameters

in	*VtdEngineNumber*	VTd engine number.

Return values

EFI_SUCCESS	Enable successfully.
EFI_INVALID_PARAMETER	Input parameters are invalid.
EFI_UNSUPPORTED	VTd base is zero.
EFI_OUT_OF_RESOURCES	There is no additional space in the PPI database.

Definition at line 72 of file VtdConfig.h.

13.71 WatchDogConfig.h File Reference

WatchDog policy.

Classes

- struct PCH_WDT_PREMEM_CONFIG

  This policy clears status bits and disable watchdog, then lock the WDT registers.

13.71.1 Detailed Description

WatchDog policy.

Copyright

INTEL CONFIDENTIAL Copyright 2015 - 2020 Intel Corporation.

Specification Reference:

Index