Client Silicon Package

v0.2.0.0

Generated by Doxygen 1.8.10

Fri May 6 2016 16:31:25

Contents

Chapter 1

1 - Introduction

Purpose

The purpose of this document is to describe the external architecture and interfaces provided in the Client Silicon Package.

Chapter 2

2 - Client Silicon Package Override (CSPO)

Client Silicon Package Override (CSPO) tags lists the overrides of the Intel Green H to add / enhance new features or resolve issues to unblock the enabling.

The Active CSPO table lists every issue which is currently resolved with an override. It is a catalog of all such existing overrides.

The Retired CSPO table lists issues which previously required such overrides. It is purely historical as these overrides have all been deleted (are no longer needed).

Active CSPOs in ClientSiliconPkg/Override

Tag	Client HSD	Core HSD	Status / Planned EOL	Description
CSPO-xxxx				

Retired CSPOs in ClientSiliconPkg/Override

Tag	Client HSD	Core HSD	Actual EOL	Description
CSPO-0001	1504011433			"Enable GCC Link Time Optimization (LTO)" to reduce the GCC build size

Chapter 3

Class Documentation

3.1 _CONFIG_BLOCK Struct Reference

Config Block.

`#include <ConfigBlock.h>`

Public Attributes

- CONFIG_BLOCK_HEADER Header

  Offset 0-27 Header of config block.

3.1.1 Detailed Description

Config Block.

Definition at line 39 of file ConfigBlock.h.

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

- ConfigBlock.h

3.2 _CONFIG_BLOCK_HEADER Struct Reference

Config Block Header.

`#include <ConfigBlock.h>`

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.

3.2.1 Detailed Description

Config Block Header.

Definition at line 29 of file ConfigBlock.h.

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

- ConfigBlock.h

3.3 _CONFIG_BLOCK_TABLE_STRUCT Struct Reference

Config Block Table Header.

`#include <ConfigBlock.h>`

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.

3.3.1 Detailed Description

Config Block Table Header.

Definition at line 49 of file ConfigBlock.h.

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

- ConfigBlock.h

3.4 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.

3.4.1 Detailed Description

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

All HOBs must contain this generic HOB header.

Definition at line 41 of file PiHob.h.

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

- PiHob.h

3.5 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>`

Public Attributes

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

3.5.1 Detailed Description

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

Definition at line 343 of file PiHob.h.

3.5.2 Member Data Documentation

3.5.2.1 EFI_HOB_GENERIC_HEADER EFI_HOB_GUID_TYPE::Header

The HOB generic header.

Header.HobType = EFI_HOB_TYPE_GUID_EXTENSION.

Definition at line 347 of file PiHob.h.

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

- PiHob.h

3.6 EFI_IP_ADDRESS Union Reference

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

`#include <UefiBaseType.h>`

3.6.1 Detailed Description

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

An IPv4 or IPv6 internet protocol address.

Definition at line 112 of file UefiBaseType.h.

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

- UefiBaseType.h

3.7 EFI_IPv4_ADDRESS Struct Reference

4-byte buffer.

`#include <UefiBaseType.h>`

3.7.1 Detailed Description

4-byte buffer.

An IPv4 internet protocol address.

Definition at line 90 of file UefiBaseType.h.

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

- UefiBaseType.h

3.8 EFI_IPv6_ADDRESS Struct Reference

16-byte buffer.

`#include <UefiBaseType.h>`

3.8.1 Detailed Description

16-byte buffer.

An IPv6 internet protocol address.

Definition at line 97 of file UefiBaseType.h.

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

- UefiBaseType.h

3.9 EFI_MAC_ADDRESS Struct Reference

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

`#include <UefiBaseType.h>`

3.9.1 Detailed Description

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

Definition at line 104 of file UefiBaseType.h.

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

- UefiBaseType.h

3.10 EFI_TABLE_HEADER Struct Reference

Data structure that precedes all of the standard EFI table types.

`#include <UefiMultiPhase.h>`

Public Attributes

- UINT64 Signature

  A 64-bit signature that identifies the type of table that follows.
- UINT32 Revision

  The revision of the EFI Specification to which this table conforms.
- UINT32 HeaderSize

  The size, in bytes, of the entire table including the EFI_TABLE_HEADER.
- UINT32 CRC32

  The 32-bit CRC for the entire table.
- UINT32 Reserved

  Reserved field that must be set to 0.

3.10.1 Detailed Description

Data structure that precedes all of the standard EFI table types.

Definition at line 129 of file UefiMultiPhase.h.

3.10.2 Member Data Documentation

3.10.2.1 UINT32 EFI_TABLE_HEADER::CRC32

The 32-bit CRC for the entire table.

This value is computed by setting this field to 0, and computing the 32-bit CRC for HeaderSize bytes.

Definition at line 151 of file UefiMultiPhase.h.

3.10.2.2 UINT32 EFI_TABLE_HEADER::Revision

The revision of the EFI Specification to which this table conforms.

The upper 16 bits of this field contain the major revision value, and the lower 16 bits contain the minor revision value. The minor revision values are limited to the range of 00..99.

Definition at line 142 of file UefiMultiPhase.h.

3.10.2.3 UINT64 EFI_TABLE_HEADER::Signature

A 64-bit signature that identifies the type of table that follows.

Unique signatures have been generated for the EFI System Table, the EFI Boot Services Table, and the EFI Runtime Services Table.

Definition at line 135 of file UefiMultiPhase.h.

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

- UefiMultiPhase.h

3.11 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>`

3.11.1 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 72 of file UefiBaseType.h.

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

- UefiBaseType.h

3.12 EFI_VARIABLE_AUTHENTICATION Struct Reference

AuthInfo is a WIN_CERTIFICATE using the wCertificateType WIN_CERTIFICATE_UEFI_GUID and the CertType EFI_CERT_TYPE_RSA2048_SHA256_GUID.

`#include <UefiMultiPhase.h>`

Public Attributes

- UINT64 MonotonicCount

  Included in the signature of AuthInfo.Used to ensure freshness/no replay.
- WIN_CERTIFICATE_UEFI_GUID AuthInfo

  Provides the authorization for the variable access.

3.12.1 Detailed Description

AuthInfo is a WIN_CERTIFICATE using the wCertificateType WIN_CERTIFICATE_UEFI_GUID and the CertType EFI_CERT_TYPE_RSA2048_SHA256_GUID.

If the attribute specifies authenticated access, then the Data buffer should begin with an authentication descriptor prior to the data payload and DataSize should reflect the the data.and descriptor size. The caller shall digest the Monotonic Count value and the associated data for the variable update using the SHA-256 1-way hash algorithm. The ensuing the 32-byte digest will be signed using the private key associated w/ the public/private 2048-bit RSA key-pair. The WIN_CERTIFICATE shall be used to describe the signature of the Variable data *Data. In addition, the signature will also include the MonotonicCount value to guard against replay attacks.

Definition at line 192 of file UefiMultiPhase.h.

3.12.2 Member Data Documentation

3.12.2.1 WIN_CERTIFICATE_UEFI_GUID EFI_VARIABLE_AUTHENTICATION::AuthInfo

Provides the authorization for the variable access.

It is a signature across the variable data and the Monotonic Count value. Caller uses Private key that is associated with a public key that has been provisioned via the key exchange.

Definition at line 208 of file UefiMultiPhase.h.

3.12.2.2 UINT64 EFI_VARIABLE_AUTHENTICATION::MonotonicCount

Included in the signature of AuthInfo.Used to ensure freshness/no replay.

Incremented during each "Write" access.

Definition at line 199 of file UefiMultiPhase.h.

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

- UefiMultiPhase.h

3.13 EFI_VARIABLE_AUTHENTICATION_2 Struct Reference

When the attribute EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS is set, then the Data buffer shall begin with an instance of a complete (and serialized) EFI_VARIABLE_AUTHENTICATION_2 descriptor.

`#include <UefiMultiPhase.h>`

Public Attributes

- EFI_TIME TimeStamp

  For the TimeStamp value, components Pad1, Nanosecond, TimeZone, Daylight and Pad2 shall be set to 0.
- WIN_CERTIFICATE_UEFI_GUID AuthInfo

  Only a CertType of EFI_CERT_TYPE_PKCS7_GUID is accepted.

3.13.1 Detailed Description

When the attribute EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS is set, then the Data buffer shall begin with an instance of a complete (and serialized) EFI_VARIABLE_AUTHENTICATION_2 descriptor.

The descriptor shall be followed by the new variable value and DataSize shall reflect the combined size of the descriptor and the new variable value. The authentication descriptor is not part of the variable data and is not returned by subsequent calls to GetVariable().

Definition at line 219 of file UefiMultiPhase.h.

3.13.2 Member Data Documentation

3.13.2.1 EFI_TIME EFI_VARIABLE_AUTHENTICATION_2::TimeStamp

For the TimeStamp value, components Pad1, Nanosecond, TimeZone, Daylight and Pad2 shall be set to 0.

This means that the time shall always be expressed in GMT.

Definition at line 224 of file UefiMultiPhase.h.

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

- UefiMultiPhase.h

3.14 FIRMWARE_VERSION Struct Reference

Firmware Version Structure.

`#include <FirmwareVersionInfoHob.h>`

3.14.1 Detailed Description

Firmware Version Structure.

Definition at line 26 of file FirmwareVersionInfoHob.h.

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

- FirmwareVersionInfoHob.h

3.15 FIRMWARE_VERSION_INFO Struct Reference

Firmware Version Information Structure.

`#include <FirmwareVersionInfoHob.h>`

Public Attributes

- UINT8 ComponentNameIndex

  Offset 0 Index of Component Name.
- UINT8 VersionStringIndex

  Offset 1 Index of Version String.
- FIRMWARE_VERSION Version

  Offset 2-6 Firmware version.

3.15.1 Detailed Description

Firmware Version Information Structure.

Definition at line 36 of file FirmwareVersionInfoHob.h.

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

- FirmwareVersionInfoHob.h

3.16 FIRMWARE_VERSION_INFO_HOB Struct Reference

Firmware Version Information HOB Structure.

`#include <FirmwareVersionInfoHob.h>`

Public Attributes

- EFI_HOB_GUID_TYPE Header

  Offset 0-23 The header of FVI HOB.
- UINT8 Count

  Offset 24 Number of FVI elements included.

3.16.1 Detailed Description

Firmware Version Information HOB Structure.

Definition at line 45 of file FirmwareVersionInfoHob.h.

3.16.2 Member Data Documentation

3.16.2.1 UINT8 FIRMWARE_VERSION_INFO_HOB::Count

Offset 24 Number of FVI elements included.

Definition at line 47 of file FirmwareVersionInfoHob.h.

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

- FirmwareVersionInfoHob.h

3.17 SMBIOS_CACHE_INFO Struct Reference

SMBIOS Cache Info HOB Structure.

`#include <SmbiosCacheInfoHob.h>`

Public Attributes

- 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.

3.17.1 Detailed Description

SMBIOS Cache Info HOB Structure.

Definition at line 32 of file SmbiosCacheInfoHob.h.

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

- SmbiosCacheInfoHob.h

3.18 SMBIOS_PROCESSOR_INFO Struct Reference

SMBIOS Processor Info HOB Structure.

`#include <SmbiosProcessorInfoHob.h>`

Public Attributes

- UINT8 ProcessorType

  ENUM defined in SMBIOS Spec v3.0 Section 7.5.1.
- UINT16 ProcessorFamily

  This info is used for both ProcessorFamily and ProcessorFamily2 fields See ENUM defined in SMBIOS Spec v3.0 Section 7.5.2.
- 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.
- UINT16 CoreCount

  This info is used for both CoreCount & CoreCount2 fields See detailed description in SMBIOS Spec v3.0 Section 7.5.6.
- UINT16 EnabledCoreCount

  This info is used for both CoreEnabled & CoreEnabled2 fields See detailed description in SMBIOS Spec v3.0 Section 7.5.7.
- UINT16 ThreadCount

  This info is used for both ThreadCount & ThreadCount2 fields See detailed description in SMBIOS Spec v3.0 Section 7.5.8.
- UINT16 ProcessorCharacteristics

  Format defined in SMBIOS Spec v3.0 Section 7.5.9.

3.18.1 Detailed Description

SMBIOS Processor Info HOB Structure.

Definition at line 32 of file SmbiosProcessorInfoHob.h.

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

- SmbiosProcessorInfoHob.h

Chapter 4

File Documentation

4.1 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>
```

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.

4.1.1 Detailed Description

Header file for Config Block Lib implementation.

THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

4.2 ConfigBlockLib.h File Reference

Header file for Config Block Lib implementation.

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.

4.2.1 Detailed Description

Header file for Config Block Lib implementation.

Copyright (c) 2015, Intel Corporation. All rights reserved.

This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at http://opensource.org/licenses/bsd-license.php

THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.

4.2.2 Function Documentation

4.2.2.1 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

4.2.2.2 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	*ConfigBlock↩ TableAddress*	- On return, points to a pointer to the beginning of Config Block Table Address

Return values

EFI_INVALID_PARAMET↩ ER	- Invalid Parameter
EFI_OUT_OF_RESOUR↩ CES	- Out of resources
EFI_SUCCESS	- Successfully created Config Block Table at ConfigBlockTableAddress

4.2.2.3 EFI_STATUS GetConfigBlock (IN VOID * *ConfigBlockTableAddress,* IN EFI_GUID * *ConfigBlockGuid,* OUT VOID ** *ConfigBlockAddress*)

Retrieve a specific Config Block data by GUID.

Parameters

in	*ConfigBlock↩ TableAddress*	- A pointer to the beginning of Config Block Table Address
in	*ConfigBlockGuid*	- A pointer to the GUID uses to search specific Config Block
out	*ConfigBlock↩ Address*	- 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

4.3 DoxygenClientSilicon.h File Reference

This file contains doxygen commands definitions for creating ClientCommonPkg documentation.

4.3.1 Detailed Description

This file contains doxygen commands definitions for creating ClientCommonPkg documentation.

Copyright

Specification

4.4 DoxygenOverride.h File Reference

This file contains doxygen commands definitions for creating Platform override documentation.

4.4.1 Detailed Description

This file contains doxygen commands definitions for creating Platform override documentation.

Copyright

4.5 FirmwareVersionInfoHob.h File Reference

Header file for Firmware Version Information.

```
#include <Uefi.h>
#include <Pi/PiHob.h>
```

Classes

- struct FIRMWARE_VERSION

  Firmware Version Structure.
- struct FIRMWARE_VERSION_INFO

  Firmware Version Information Structure.
- struct FIRMWARE_VERSION_INFO_HOB

  Firmware Version Information HOB Structure.

4.5.1 Detailed Description

Header file for Firmware Version Information.

4.6 Fit.h File Reference

This file contains definitions for FIT table entries including error string definitions.

4.6.1 Detailed Description

This file contains definitions for FIT table entries including error string definitions.

Copyright

Copyright (c) 2015 - 2016 Intel Corporation. All rights reserved This software and associated documentation (if any) is furnished under a license and may only be used or copied in accordance with the terms of the license. Except as permitted by the license, no part of this software or documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation. This file contains 'Framework Code' and is licensed as such under the terms of your license agreement with Intel or your vendor. This file may not be modified, except as allowed by additional terms of your license agreement.

Specification

4.7 HstiFeatureBit.h File Reference

This file contains various definitions for IHV HSTI implementation including error string definitions.

4.7.1 Detailed Description

This file contains various definitions for IHV HSTI implementation including error string definitions.

Copyright

Copyright (c) 2015 - 2016 Intel Corporation. All rights reserved This software and associated documentation (if any) is furnished under a license and may only be used or copied in accordance with the terms of the license. Except as permitted by the license, no part of this software or documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation. This file contains 'Framework Code' and is licensed as such under the terms of your license agreement with Intel or your vendor. This file may not be modified, except as allowed by additional terms of your license agreement.

Specification

4.8 PiBootMode.h File Reference

Present the boot mode values in PI.

Typedefs

- typedef UINT32 EFI_BOOT_MODE

  EFI boot mode.

4.8.1 Detailed Description

Present the boot mode values in PI.

Revision Reference:

PI Version 1.2.1A

4.9 PiHob.h File Reference

HOB related definitions in PI.

Classes

- struct EFI_HOB_GENERIC_HEADER

  Describes the format and size of the data inside the HOB.
- struct EFI_HOB_GUID_TYPE

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

Typedefs

- typedef UINT32 EFI_RESOURCE_TYPE

  The resource type.
- typedef UINT32 EFI_RESOURCE_ATTRIBUTE_TYPE

  A type of recount attribute type.

4.9.1 Detailed Description

HOB related definitions in PI.

Revision Reference:

PI Version 1.4

4.10 SmbiosCacheInfoHob.h File Reference

Header file for SMBIOS Cache Info HOB.

```
#include <Uefi.h>
#include <Pi/PiHob.h>
```

Classes

- struct SMBIOS_CACHE_INFO

  SMBIOS Cache Info HOB Structure.

4.10.1 Detailed Description

Header file for SMBIOS Cache Info HOB.

Specification Reference:

System Management BIOS (SMBIOS) Reference Specification v3.0.0 dated 2015-Feb-12 (DSP0134) http://www.dmtf.org/sites/default/files/standards/documents/DSP0134_3.0.0.pdf

4.11 SmbiosProcessorInfoHob.h File Reference

Header file for SMBIOS Processor Info HOB.

```
#include <Uefi.h>
#include <Pi/PiHob.h>
```

Classes

- struct SMBIOS_PROCESSOR_INFO

  SMBIOS Processor Info HOB Structure.

4.11.1 Detailed Description

Header file for SMBIOS Processor Info HOB.

Specification Reference:

System Management BIOS (SMBIOS) Reference Specification v3.0.0 dated 2015-Feb-12 (DSP0134) http://www.dmtf.org/sites/default/files/standards/documents/DSP0134_3.0.0.pdf

4.12 TraceHubControlLib.h File Reference

Definition of TraceHubControlLib interface.

Functions

- UINT8 TraceHubControlRouting (VOID)

  Read from TraceHub Hardware Routing Enable/Disable bits.
- UINT32 TraceHubControlDebugLevel (VOID)

  Read from TraceHub Hardware Debug Level bits.

4.12.1 Detailed Description

Definition of TraceHubControlLib interface.

4.12.2 Function Documentation

4.12.2.1 UINT32 TraceHubControlDebugLevel (VOID)

Read from TraceHub Hardware Debug Level bits.

Return values

Debug	Level Bits value.

4.12.2.2 UINT8 TraceHubControlRouting (VOID)

Read from TraceHub Hardware Routing Enable/Disable bits.

Return values

Routing	Enable/Disable bits value.

4.13 TraceHubDebugExLib.h File Reference

Definition of TraceHubDebugExLib interface.

Functions

- RETURN_STATUS TraceHubDebugWriteEx (IN UINTN DebugLevel, IN UINT8 *Buffer, IN UINTN Number↩ OfBytes)

  Check if TraceHub control blocks this write.

4.13.1 Detailed Description

Definition of TraceHubDebugExLib interface.

4.13.2 Function Documentation

4.13.2.1 RETURN_STATUS TraceHubDebugWriteEx (IN UINTN *DebugLevel,* IN UINT8 * *Buffer,* IN UINTN *NumberOfBytes*)

Check if TraceHub control blocks this write.

Parameters

DebugLevel	The debug level of the debug message.
Buffer	Pointer to the data buffer to be written.
NumberOfBytes	Number of bytes to written to TraceHub device.

Return values

RETURN_SUCCESS	Success to write the buffer to TraceHub.
RETURN_ABORTED	Aborted due to TraceHub control.
Other	Fail to write the buffer to TraceHub.

4.14 UefiBaseType.h File Reference

Defines data types and constants introduced in UEFI.

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

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_IPv4_ADDRESS

  4-byte buffer.
- struct EFI_IPv6_ADDRESS

  16-byte buffer.
- 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_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.

4.14.1 Detailed Description

Defines data types and constants introduced in UEFI.

4.14.2 Macro Definition Documentation

4.14.2.1 #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 219 of file UefiBaseType.h.

4.14.2.2 #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 206 of file UefiBaseType.h.

4.15 UefiMultiPhase.h File Reference

This includes some definitions introduced in UEFI that will be used in both PEI and DXE phases.

```
#include <Guid/WinCertificate.h>
```

Classes

- struct EFI_TABLE_HEADER

Data structure that precedes all of the standard EFI table types.

- struct EFI_VARIABLE_AUTHENTICATION

  AuthInfo is a WIN_CERTIFICATE using the wCertificateType WIN_CERTIFICATE_UEFI_GUID and the CertType EFI_CERT_TYPE_RSA2048_SHA256_GUID.

- struct EFI_VARIABLE_AUTHENTICATION_2

  When the attribute EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS is set, then the Data buffer shall begin with an instance of a complete (and serialized) EFI_VARIABLE_AUTHENTICATION_2 descriptor.

Macros

- #define EFI_VARIABLE_NON_VOLATILE 0x00000001

  Attributes of variable.

- #define EFI_VARIABLE_HARDWARE_ERROR_RECORD 0x00000008

  This attribute is identified by the mnemonic 'HR' elsewhere in this specification.

- #define EFI_VARIABLE_AUTHENTICATED_WRITE_ACCESS 0x00000010

  Attributes of Authenticated Variable.

Enumerations

- enum EFI_MEMORY_TYPE

  Enumeration of memory types introduced in UEFI.

- enum EFI_RESET_TYPE

  Enumeration of reset types.

4.15.1 Detailed Description

This includes some definitions introduced in UEFI that will be used in both PEI and DXE phases.

4.15.2 Enumeration Type Documentation

4.15.2.1 enum EFI_MEMORY_TYPE

Enumeration of memory types introduced in UEFI.

Enumerator

EfiReservedMemoryType Not used.

EfiLoaderCode The code portions of a loaded application. (Note that UEFI OS loaders are UEFI applications.)

EfiLoaderData The data portions of a loaded application and the default data allocation type used by an application to allocate pool memory.

EfiBootServicesCode The code portions of a loaded Boot Services Driver.

EfiBootServicesData The data portions of a loaded Boot Serves Driver, and the default data allocation type used by a Boot Services Driver to allocate pool memory.

EfiRuntimeServicesCode The code portions of a loaded Runtime Services Driver.

EfiRuntimeServicesData The data portions of a loaded Runtime Services Driver and the default data allocation type used by a Runtime Services Driver to allocate pool memory.

EfiConventionalMemory Free (unallocated) memory.

EfiUnusableMemory Memory in which errors have been detected.

EfiACPIReclaimMemory Memory that holds the ACPI tables.

EfiACPIMemoryNVS Address space reserved for use by the firmware.

EfiMemoryMappedIO Used by system firmware to request that a memory-mapped IO region be mapped by the OS to a virtual address so it can be accessed by EFI runtime services.

EfiMemoryMappedIOPortSpace System memory-mapped IO region that is used to translate memory cycles to IO cycles by the processor.

EfiPalCode Address space reserved by the firmware for code that is part of the processor.

EfiPersistentMemory A memory region that operates as EfiConventionalMemory, however it happens to also support byte-addressable non-volatility.

Definition at line 22 of file UefiMultiPhase.h.

4.15.2.2 enum EFI_RESET_TYPE

Enumeration of reset types.

Enumerator

EfiResetCold Used to induce a system-wide reset. This sets all circuitry within the system to its initial state. This type of reset is asynchronous to system operation and operates withgout regard to cycle boundaries. EfiColdReset is tantamount to a system power cycle.

EfiResetWarm Used to induce a system-wide initialization. The processors are set to their initial state, and pending cycles are not corrupted. If the system does not support this reset type, then an EfiResetCold must be performed.

EfiResetShutdown Used to induce an entry into a power state equivalent to the ACPI G2/S5 or G3 state. If the system does not support this reset type, then when the system is rebooted, it should exhibit the EfiResetCold attributes.

EfiResetPlatformSpecific Used to induce a system-wide reset. The exact type of the reset is defined by the EFI_GUID that follows the Null-terminated Unicode string passed into ResetData. If the platform does not recognize the EFI_GUID in ResetData the platform must pick a supported reset type to perform. The platform may optionally log the parameters from any non-normal reset that occurs.

Definition at line 96 of file UefiMultiPhase.h.

Index