AMD Generic Encapsulated Software Architecture (*AGESA™*) Interface Specification for Arch2008

Publication # **44065**	Revision: **3.00**
Issue Date: **July 2014**	

Specification Agreement

This Specification Agreement (this "Agreement") is a legal agreement between Advanced Micro Devices, Inc. ("AMD") and "You" as the recipient of the attached AMD Specification (the "Specification"). If you are accessing the Specification as part of your performance of work for another party, you acknowledge that you have authority to bind such party to the terms and conditions of this Agreement. If you accessed the Specification by any means or otherwise use or provide Feedback (defined below) on the Specification, You agree to the terms and conditions set forth in this Agreement. If You do not agree to the terms and conditions set forth in this Agreement, you are not licensed to use the Specification; do not use, access or provide Feedback about the Specification.

In consideration of Your use or access of the Specification (in whole or in part), the receipt and sufficiency of which are acknowledged, You agree as follows:

1. You may review the Specification only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with an AMD product in compliance with the requirements as set forth in the Specification and (b) to provide Feedback about the information disclosed in the Specification to AMD.

2. Except as expressly set forth in Paragraph 1, all rights in and to the Specification are retained by AMD. This Agreement does not give You any rights under any AMD patents, copyrights, trademarks or other intellectual property rights. You may not (i) duplicate any part of the Specification; (ii) remove this Agreement or any notices from the Specification, or (iii) give any part of the Specification, or assign or otherwise provide Your rights under this Agreement, to anyone else.

3. The Specification may contain preliminary information, errors, or inaccuracies, or may not include certain necessary information. Additionally, AMD reserves the right to discontinue or make changes to the Specification and its products at any time without notice. The Specification is provided entirely "AS IS." AMD MAKES NO WARRANTY OF ANY KIND AND DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, TITLE OR THOSE WARRANTIES ARISING AS A COURSE OF DEALING OR CUSTOM OF TRADE. AMD SHALL NOT BE LIABLE FOR DIRECT, INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE OR EXEMPLARY DAMAGES OF ANY KIND (INCLUDING LOSS OF BUSINESS, LOSS OF INFORMATION OR DATA, LOST PROFITS, LOSS OF CAPITAL, LOSS OF GOODWILL) REGARDLESS OF THE FORM OF ACTIONWHETHER IN CONTRACT, TORT (INCLUDING NEGLIGENCE) AND STRICT PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4. Furthermore, AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur.

5. You have no obligation to give AMD any suggestions, comments or feedback ("Feedback") relating to the Specification. However, any Feedback You voluntarily provide may be used by AMD without restriction, fee or obligation of confidentiality. Accordingly, if You do give AMD Feedback on any version of the Specification, You agree AMD may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any product, as well as has the right to sublicense third parties to do the same. Further, You will not give AMD any Feedback that You may have reason to believe is (i) subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any product or intellectual property incorporating or derived from Feedback or any Product or other AMD intellectual property to be licensed to or otherwise provided to any third party.

6. You shall adhere to all applicable U.S., European, and other export laws, including but not limited to the U.S. Export Administration Regulations ("EAR"), (15 C.F.R. Sections 730 through 774), and E.U. Council Regulation (EC) No 428/2009 of 5 May 2009. Further, pursuant to Section 740.6 of the EAR, You hereby certifies that, except pursuant to a license granted by the United States Department of Commerce Bureau of Industry and Security or as otherwise permitted pursuant to a License Exception under the U.S. Export Administration Regulations ("EAR"), You will not (1) export, re-export or release to a national of a country in Country Groups D:1, E:1 or E:2 any restricted technology, software, or source code You receive hereunder, or (2) export to Country Groups D:1, E:1 or E:2 the direct product of such technology or software, if such foreign produced direct product is subject to national security controls as identified on the Commerce Control List (currently found in Supplement 1 to Part 774 of EAR). For the most current Country Group listings, or for additional information about the EAR or Your obligations under those regulations, please refer to the U.S. Bureau of Industry and Security's website at http://www.bis.doc.gov/.

7. If You are a part of the U.S. Government, then the Specification is provided with "RESTRICTED RIGHTS" as set forth in subparagraphs (c) (1) and (2) of the Commercial Computer Software-Restricted Rights clause at FAR 52.227-14 or subparagraph (c) (1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013, as applicable.

8. This Agreement is governed by the laws of the State of California without regard to its choice of law principles. Any dispute involving it must be brought in a court having jurisdiction of such dispute in Santa Clara County, California, and You waive any defenses and rights allowing the dispute to be litigated elsewhere. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. The failure of AMD to enforce any rights granted hereunder or to take action against You in the event of any breach hereunder shall not be deemed a waiver by AMD as to subsequent enforcement of rights or subsequent actions in the event of future breaches. This Agreement is the entire agreement between You and AMD concerning the Specification; it may be changed only by a written document signed by both You and an authorized representative of AMD.

Disclaimer

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks

AMD, the AMD Arrow logo, AGESA, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

HDMI is a trademark of HDMI Licensing, LLC.

HyperTransport is a licensed trademark of the HyperTransport Technology Consortium.

Microsoft is a registered trademarks of Microsoft Corporation.

PCIe is a registered trademark of PCI-Special Interest Group (PCI-SIG).

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Dolby Laboratories, Inc.

Manufactured under license from Dolby Laboratories.

Rovi Corporation

This device is protected by U.S. patents and other intellectual property rights. The use of Rovi Corporation's copy protection technology in the device must be authorized by Rovi Corporation and is intended for home and other limited pay-per-view uses only, unless otherwise authorized in writing by Rovi Corporation.

Reverse engineering or disassembly is prohibited.

USE OF THIS PRODUCT IN ANY MANNER THAT COMPLIES WITH THE MPEG ACTUAL OR DE FACTO VIDEO AND/OR AUDIO STANDARDS IS EXPRESSLY PROHIBITED WITHOUT ALL NECESSARY LICENSES UNDER APPLICABLE PATENTS. SUCH LICENSES MAY BE ACQUIRED FROM VARIOUS THIRD PARTIES INCLUDING, BUT NOT LIMITED TO, IN THE MPEG PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, L.L.C., 6312 S. FIDDLERS GREEN CIRCLE, SUITE 400E, GREENWOOD VILLAGE, COLORADO 80111.

AGESA™ Software Fundamentals

Porting Guide for Binary Module

Porting Guide for UEFI

API Specification

Integrated Debug Services (IDS)

Appendices

List of Figures

Revision History

Date	Revision	Description
July 2014	v3.00	Initial public release

Section I - AGESA™ Software Fundamentals

Chapter 1 Introduction

The AMD Generic Encapsulated Software Architecture (AGESA™) software is a BIOS procedure library designed to aid AMD customers to quickly implement AMD technology into their products.

This document covers the interface definition for the procedure library and provides some guidelines on how to use the library in the customer's environment. The library is designed to support multiple AMD products. This chapter explains the goals of the AGESA™ software.

1.1 Goals

The goal of this document is to provide a user-level description of the procedures and abilities of the AGESA™ software library.

1.2 Audience

This document is directed to the host environment wrapper implementer. It defines procedure call interfaces, specifies a recommended wrapper implementation and describes options that the wrapper implementer may choose to use. Some knowledge of BIOS programming practices is required.

1.3 Related Documents

Following is a list of related AMD documents:

- BIOS and Kernel Developer's Guides (BKDG)
  - *BIOS and Kernel Developer's Guide for AMD Family 15h Models 00h-0Fh Processors*, order# 42301
  - *BIOS and Kernel Developer's Guide for AMD Family 15h Models 10h-1Fh Processors*, order# 42300
  - *BIOS and Kernel Developer's Guide for AMD Family 15h Models 30h-3Fh Processors*, order# 49125
- Revision Guides
  - *Revision Guide for AMD Family 15h Models 00h-0Fh Processors,* order# 48063
  - *Revision Guide for AMD Family 15h Models 10h-1Fh Processors*, order# 48931

- *Infrastructure Roadmap (IRM) Specifications*
  - FT3b Infrastructure Roadmap (NDA), order #53081
  - FS1b Infrastructure Roadmap (NDA), order #52171
  - FP3 Infrastructure Roadmap (NDA), order #50819
  - FT3 Infrastructure Roadmap (NDA), order #50818
- *Thermal Design for Fanless Tablets*, order #53195
- *CPUID Specification*, order# 25481
- *Socket G34 Motherboard Design Guide*, order# 45934
- *Socket C32 Processor Motherboard Design Guide*, order# 47394

Note: Please note for Socket G34 and Socket C32, that the attachment of the SA[2:0] pins in a unique and consecutive numeric order is required by the AGESA™ software for proper identification of sockets. These need to be connected regardless of whether the platform supports APML or not. All of the processors must be addressed on the same APML bus.

- *Socket G34 Processor Functional Data Sheet*, order# 45937
- *Socket C32 Processor Functional Data Sheet*, order# 47390
- *FS1 Processor Motherboard Design Guide*, order# 44730
- *Socket FS1 Processor Functional Data Sheet*, order# 44086
- *FM1 Processor Motherboard Design Guide*, order# 47757
- *Socket FM1 Processor Functional Data Sheet*, order# 44085
- *FP1 Processor Motherboard Design Guide*, order# 47760
- *FP1F Processor Functional Data Sheet*, order# 47764
- *Socket FS2 Processor Motherboard Design Guide*, order# 50578
- *Socket FS2 Processor Functional Data Sheet*, order# 51237
- *FP3 Processor Motherboard Design Guide*, order# 50796
- *FP3 Processor Functional Data Sheet*, order# 51235
- *Socket AM3 Motherboard Design Guide*, order# 40837
- *Socket AM3 Processor Functional Data Sheet*, order# 40778

Related industry standards documents:

- DDR2 SDRAM Specification (JESD79-2), November 2004.
- PC2-6400/PC2-5300/PC2-4200/PC2-3200 Registered DIMM Design Specification
- PC2-5300/PC2-6400 DDR2 SDRAM Unbuffered DIMM Design Specification
- Appendix X: SPD for DDR2 SDRAM Module
- Advanced Configuration and Power Interface Specification, (ACPI) revision 4.0
- System Management BIOS (SMBIOS) Reference Specification, DMTF DSP0134, Version 2.6a

- MultiProcessor Specification (MP Spec), Intel corp., version 1.4
- C Coding Standards Specification, UEFI consortium, USWG, v0.3

1.4 Terminology

The following definitions, acronyms and terms are used in this specification:

Host Environment	The main body of code responsible for power-up sequencing and initialization of the platform. This can be one of many code bases in today's technical markets.
Link	A HyperTransport™ technology connection between two devices.
MCT	Name used to refer to the memory controller and the memory controller initialization code.
Node	Refers to a processor or other device actively participating in the HyperTransport link coherent fabric.
Socket	Physical connector on the motherboard into which a processor can be installed. Sockets are numbered relative to the system or motherboard and are software-identified by the SA[2:0] pins.
Processor	AMD silicon product which fits into a socket and may contain one or more die.
Die	With the advent of Multi-Chip-Modules (MCM), a single processor may contain more than one silicon wafer die. Each die may contain one or more cores. Die are numbered relative to the processor.
Core	A term used to refer to the portion of a central processing unit which executes computational programs. Software sees each core as a separate entity even though two or more cores may physically reside in the same processor, or in the same die. Cores are numbered relative to the processor regardless of the die on which they may be physically located.
System address	Physical 40-bit address (without translations).
Wrapper	The layer of code used to isolate the AGESA™ software code from the host architecture code.
MP - Multi-Processor or Multi-Processing	This refers to a programming environment in which multiple processor cores are executing code simultaneously. In such an environment, special programming practices must be used to communicate between processor cores and to synchronize their operations.
BSP - Boot Strap Processor	In an MP environment this is the master processor core. This core controls what operations are allocated to the other cores.
AP - Application Processor	In an MP environment this is a slave processor core that performs operations as directed by the BSP. Once the operation

	is complete, the core stops execution and waits for its next instructions from the BSP.
<Plat>Options	This is a general term convention to indicate an item that is specific to a Platform Solution. For example, when applied to a file, "<Plat>Options.c" would be interpreted as a file having the name "MaranelloOptions.c"
BKDG	A short hand reference to the *BIOS and Kernel Developer's Guides (BKDG)*

Chapter 2 Logistical Matters

2.1 Version Numbering

The AGESA™ software uses a three-level version numbering scheme: X.Y.Z. This interface specification is updated along with the code and the specification version tracks with the code version except that the specification only uses a two-level version numbering scheme, X.Y, where:

X - Major version number	A major version number of 0 indicates a prototype status. The first production ready version has a major version number of 1, then increments for further changes. This number increments when a major new feature or feature set is added.
Y - Minor version number	This number is incremented when a minor change occurs in the interface specification.Example of a minor change is the addition of a new function or feature.
Z - Code update number	This number is incremented for each software release. Changes at this level indicate no change has occurred in the interface. Code changes are bug fixes only. Note that this number may be more than one digit long.

Incrementing a version sub-number causes all lower-level sub-numbers to revert to 0. The product name is not part of the version numbering, but it can be used to designate the full title. Take “AGESA MarG34PI V1.2.3” for example: the “AGESA MarG34PI” is the product title and the “V1.2.3” is the version number.

The version string is embedded into the code image and can be discovered by a program. The program searches the ROM image for the signature in the following structure. Once found, the program can start at the 3rd byte and use the rest of the structure as a null terminated ASCII string.

```
// AMD_CODE_HEADER Signatures.
#define AGESA_CODE_SIGNATURE  {'!', '!', 'A', 'G', 'E', 'S', 'A', ' '}
#define CIMXNB_CODE_SIGNATURE {'!', '!', 'C', 'I', 'M', 'X', 'N', 'B'}
#define CIMXSB_CODE_SIGNATURE {'!', '!', 'C', 'I', 'M', 'X', 'S', 'B'}

/// AGESA_CODE_SIGNATURE
typedef struct {
      IN    CHAR8 Signature[8];     // code header Signature
      IN    CHAR8 ComponentName[8]; // 8 character name of the code module
      IN    CHAR8 Version[12];      // 12 character version string
      IN    CHAR8 TerminatorNull;   // = 0x00, null terminated string
      IN    CHAR8 VerReserved[7];   // reserved space
} AMD_CODE_HEADER;
```

2.2 Release Package

The release package given to customers includes the following:

AGESA™ software technology core files	Documentation (release notes)
Binary block interfacing files	UEFI interfacing files
Binary compilations (all options set ‘on’)	Build control files
Examples of user-generated files (wrapper)	Specialized build tools

The files contained in the distribution are organized into a directory hierarchy similar to the one as shown in Figure 2.1below:

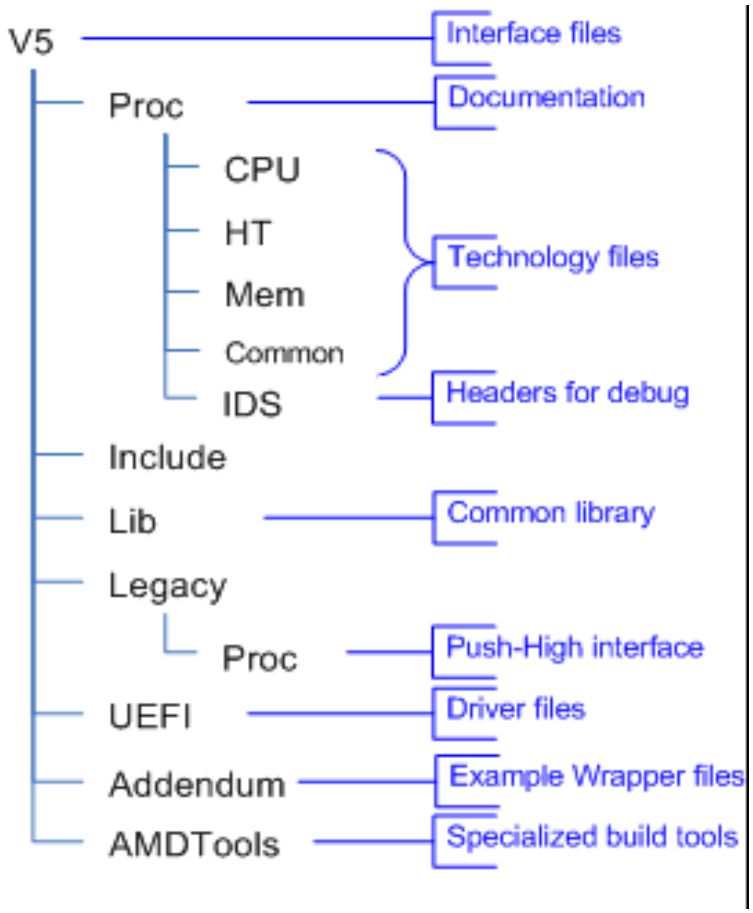

Figure 2.1: Package Directory Structure

2.3 File Naming Convention

The AGESA™ software complies with the *C Coding Standards Specification, UEFI consortium, USWG, v0.3* which states the core "C" code files use the long file name format. This removes any need for a cryptic naming convention, the file names are descriptive.

2.4 File headers

Every source file contains a comment header describing the file function and purpose, as defined in the *C Coding Standards Specification, UEFI consortium, USWG, v0.3*. The following is an example comment header as used in the AGESA™ software files:

```
/**
 * @file
 *
 * Text Title of file.
 *
 * Purpose of File Content.
 *
 * @xrefitem bom "File Content Label" "Release Content"
 * @e project:      AGESA
 * @e sub-project: Library
 * @e version: $Revision$        @e date:    $Date$
 *
```

This is immediately followed by the AMD copyright notice, which is not shown here. The revision number in the last line is the revision number of the individual file as recorded in the AMD source control system. This number has no relationship to the code block version number.

2.5 Procedure Naming Convention

All public procedures use a name prefix to assist the programmer in understanding his or her place and purpose. All call-in entry points to the AGESA™ software use a prefix: "Amd" (for example, AmdInitReset). All call-out procedures used to gather data or provide customizing opportunities use a prefix: "Agesa" (for example, AgesaReadSpd).

2.6 Internal Documentation

This document describes the user interface to which the implemented code must conform. For more detailed information about the internal design or how the sub-systems work, there is internal documentation shipped with the sources in the delivery package. This internal documentation is generated from the source files using the open source tools 'Doxygen' and 'Dot'. The HTML output is quite impressive and includes linked function and structure definitions, include dependency trees, call trees and where used lists.

2.6.1 Installing the Doxygen and Dot Tools

Doxygen is an open source, source code documentation generator tool. Documentation, user guides and downloads can be found at:

http://www.stack.nl/~dimitri/doxygen/index.html

Doxygen uses the Dot graphics generation tool to make visual diagrams and graphs of the dependency and call trees. Documentation and downloads can be found at:

http://www.graphviz.org/

2.6.2 Building the HTML files

The AGESA™ software includes a Microsoft® Visual Studio SLN control file used for building the HTML internal documentation files. The build control file for the processor documentation can be found at `\Proc\AgesaDoc.sln` within the release package.

The documentation build can be run from the command line (see the inputs for the Legacy.Bat file) or it can be run interactively by opening MS Visual Studio with the proper environment variables established (same as needed for the code build), loading the AgesaDoc.sln file, then selecting the project '**rebuild**' command. The HTML files will be placed in an Html sub-directory under the platform tip directory. An internet browser may be used to open mainpage.html in that folder.

For convenience, a single help file called *arch2008.chm* is placed at the same level as the html folder. This is the complete help output and may be used in place of the html folder content. A copy is provided within the release package.

Chapter 3 Design Concepts

3.1 Interface Design Theory

The goal for the interface is to remain a constant for several generations of processors. To accomplish this goal, the number of call entry points and the content of the interface are reduced as much as possible. The call entry points are based on the boot time line and represent time points in the boot process instead of specific technical functions. See "4.1 AGESA™ Core Software" on page 37 for more details.

3.1.1 Required versus Optional Parameters

The AGESA™ software vigorously seeks to automatically determine configuration parameters for the host system. However, there are limits of technology and practicality, which means that there are still some required parameters that the host environment must provide. These parameters are identified in the call definitions. Many other parameters are available to the host environment for tailoring the configuration, but these are optional and the host environment may modify those it chooses. If not modified, the default values determined by the AGESA™ software are used. The default settings are identified in the interface definitions.

3.1.2 Call Mechanics

To greatly simplify the host environment implementation of an AGESA™ software interface call, this architecture uses the concept of an "initializer" function. See Figure 3.2 on page 27. The initializer allocates space for the interface data structure and pre-fills the structure with default parameter settings. The host environment must then fill in any required parameters and may modify any additional parameters of interest. After the main function is called, the host environment checks error status, then releases the interface data structure.

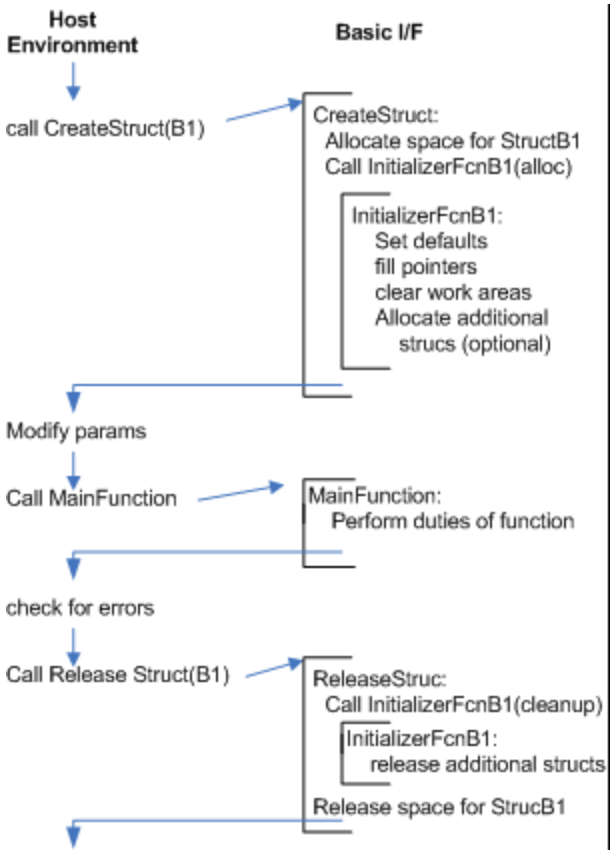

Figure 3.2: Interface Call Flow

A code example for this process is as follows:

```
VOID
OemMyProcedure (UINT8 MyParam) {
      AGESA_STATUS            MyStatus;
      CREATE_STRUCT_PARAMS    LclConfigBlk;
      AMD_EARLY_PARAMS        *MyPtr;

      /* Fill the local Config Block */
      LclConfigBlk.StdHeader.Func         = AMD_CREATE_STRUC
      LclConfigBlk.StdHeader.ImageBase    = 0xFFFE0000;
      LclConfigBlk.StdHeader.AltImageBase = 0x00000000;
      LclConfigBlk.StdHeader.PcieBasePtr  = 0x00000000;
      LclConfigBlk.StdHeader.CallBackPtr  = &OemMyCbDispatcher;
      LclConfigBlk.ParamStructName        = AMD_EARLYINIT_PARAMS;
      LclConfigBlk.AllocMethod            = PREMEM_HEAP;

      /* Call the AMD Initializer function */
      MyStatus = AmdBridge32(&LclConfigBlk);
      MyPtr = (AMD_EARLY_PARAMS *)LclConfigBlk.NewStruc;
```

```
      /* OEM has chance to change parameter struct content */
      MyPtr->EarlyParams.RandomParameter = MY_OEM_VALUE;

      /* Call the AGESA main entry point */
      MyPtr->StdHeader.Func = AMD_EARLY_INIT;
      MyStatus = AmdBridge32(MyPtr);
      /* Check return status */
      switch (MyStatus) {
      case AGESA_WARNING:
            printf("Warning msg");
            break;
      case AGESA_ERROR:
            /* handle error */
      };
      /* Call to release the data structure */
      LclConfigBlk.StdHeader.Func= AMD_RELEASE_STRUC
      MyStatus = AmdBridge32(&LclConfigBlk);
};
```

All interface structure definitions are provided in an include file.

3.1.3 Error Reporting

Two methods of error reporting are provided: return codes and error logs. The return codes are described in "4.3.1 Returned Status Codes" on page 41 and provide enough detail for most host environments. For more detailed error reporting, the error log system is provided (see section "4.3.2 Error Logging" on page 42 for more details).

3.2 Build Design Theory

All pertinent data needed to select build options and to describe the target platform is contained in one user file named "<Plat>Options.c". See Figure 3.3 on page 29. This file contains a list of #define entries that are used to conditionally compile the AGESA™ software defaults tables. It is these defaults tables that are used by the "initializer" functions to pre-fill the procedure parameter data structures prior to the call to the main function as described in "3.1.2 Call Mechanics" on page 26.

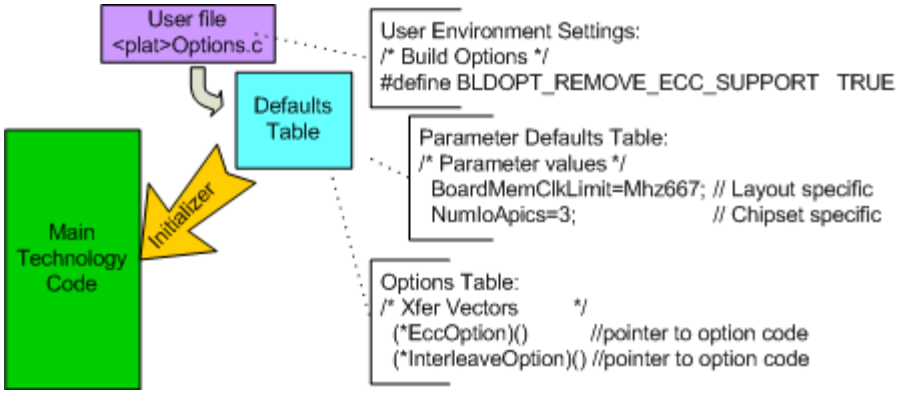

Figure 3.3: Build Design Diagram

The main technology code is compiled, without #defines and without the <Plat>Options.c file. The main technology code then becomes a code library for the platform build.

The defaults tables file is compiled separately, referencing the <Plat>Options.c file. It extensively evaluates and cross checks the user options listed and outputs a data object file containing parameter data default values and reference pointers to selected option procedures. When linked with the main technology code library, the reference points cause only the desired option procedures to be included into the build. The location of this output object file must be specified to the AGESA™ module build control files.

The <Plat>Options.c file is located in a platform-specific directory according to the host environment needs. The defaults tables output file is located in the same platform-specific directory. The host environment may choose to reference it from that location or may choose to copy or move it to another directory location.

3.2.1 Control Files

The AGESA™ software contains build control files. In sync with the defined tool set, these control files are Microsoft® Visual Studio project files. The host environment may choose to use these build control files or use them as reference to create their own. Project files are provided for both the Microsoft® Visual Studio 2005 and 2010 versions.

The control files are nested or layered to allow modular construction. The directories have a top-level control file that invokes control files from the other directories to assemble components needed for their interface.

3.2.2 Build Styles

Some of the control files are designed to create binary images that can be built separately from the host environment code and merged into the final ROM image. This creates the ability to have multiple build styles:

- **Direct to module dispatcher —** This style uses the separated binary image build, useing a direct call to the module dispatcher entry point. The host environment must parse the binary header described in the next section to locate the module dispatcher entry point.
- **Full integration with host environment —** This style does not use the binary image but pulls the AGESA™ software sources into the host environment build tools. The host environment code, using this style, may make direct calls to the AGESA™ software published functions. This style is used by the UEFI drivers.

3.3 Binary Image Design

The binary image build style is used to isolate the AGESA support code from the host environment in a way that allows the two to be compiled separately then merged as binary blocks into a final BIOS ROM image.

3.3.1 Binary Block Structure

An AGESA™ software binary image is prefixed with a binary header that identifies the binary and points to the code modules contained within. There may be only one code module or several may be combined into one binary image to reduce build overhead costs. See “3.4 Binary Image Implementation” on page 34.

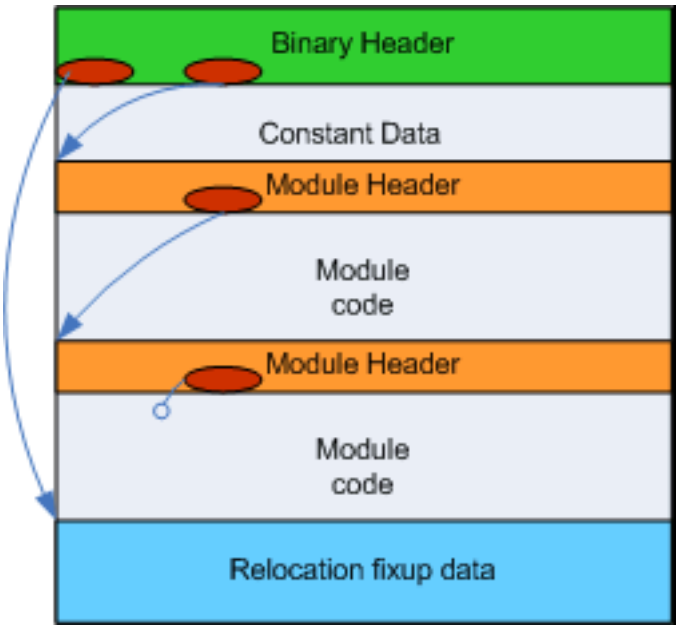

Figure 3.4: Binary Image Layout

The host environment must first find the binary image in the memory space, then calculate the execution address of the dispatcher to transfer control to the desired AGESA™ software function.

3.3.2 Execution Environment Expectations

The image code expects to be ‘located’ at an address determined by the host environment. This means the relocation fix-up records are used to fix up the code references to addresses in the image. AMD provides a tool called BINUTIL2.EXE to perform this action. Options and usage of this tool are listed in “A.1 BINUTIL2” on page 259.

Prior to calling any AGESA™ software interface, the host environment must establish the following requirements:

- CPU is in 32-bit protected mode execution
- CS is a 32-bit code segment with Base/Limit – 0x00000000/0xffffffff
- DS/ES/SS are 32-bit data segments with Base/Limit – 0x00000000/0xffffffff
- If paging is enabled, then pages must be identity-mapped (pg. 1 to pg.1, pg. 2 to pg. 2, etc.)

This configuration is commonly known as ‘32-bit Flat Address’ mode.

3.3.3 Standard Header

An architectural header is defined at the front of every function's configuration block. This standard header is filled by the host environment prior to calling through the entry point. This data is present in the configuration block at all times, essentially making it global throughout the function call.

Prototype

```
typedef struct {
    IN      UINT32            ImageBasePtr;
    IN      UINT32            Func;
    IN      UINT32            AltImageBasePtr;
    IN      CALLOUT_ENTRY     CalloutPtr;
    IN      UINT8             HeapStatus;
    IN      UINT64            HeapBasePtr;
    IN OUT  UINT8             Reserved[7];
} AMD_CONFIG_PARAMS;
```

Parameters

Func

The identifier of the desired procedure. The identifiers are defined in a provided include file.

ImageBasePtr

This is the memory location where the host environment has placed the binary image.

AltImageBasePtr

This is the memory location where the host environment has placed another secondary binary image. This value is optional and should be set to 0x00000000 when not used.

PcieBasePtr

This is the base address of the memory-mapped I/O block for access to PCIe® configuration cycles.

CalloutPtr

This is a 32-bit pointer to where the host environment has placed the entry point for call-out routines. The AGESA™ software code transfers control to this address for processing of call-out functions. See "3.4 Binary Image Implementation" on page 34.

HeapStatus

This item identifies the heap location, such as in Cache-As-Ram or memory. AmdCreateStruct will set this field when it creates the header for each entry point based on the expected heap status at that entry point. See "Figure 4.6:

AGESA Boot Time Call Points" on page 39 for a description of how the entry points fit into a boot sequence.

HeapBasePtr

This is the base address of the AGESA heap. AmdCreateStruct will set this field when it creates the header for the entry point.

Reserved

This space is reserved for future use.

3.4 Binary Image Implementation

Binary image implementation information is contained in the following sections.

3.4.1 Binary Header

The following binary header exists in each binary image. It is the first element in the binary image and can be discovered by either implicit build knowledge or by searching the memory space for the signature constant. The binary image and therefore the signature must be aligned on a 32-Kbyte boundary in the memory address space.

Prototype

```
typedef struct {
      IN    UINT32          Signature;
      IN    CHAR8           CreatorID[8];
      IN    CHAR8           Version[12];
      IN    UINT32          ModuleInfoOffset;
      IN    UINT32          EntryPointAddress;
      IN    UINT32          ImageBase;
      IN    UINT32          RelocTableOffset;
      IN    UINT32          ImageSize;
      IN    UINT16          Checksum;
      IN    UINT8           ImageType;
      IN    UINT8           V_Reserved;
} AMD_IMAGE_HEADER;
```

Parameters

Signature

Signature that identifies this as an AGESA™ software image. This is a constant value equal to the string “$AMD”.

Creator ID

Image creator ID signature. This is specified by a parameter to, and set by the BINUTI2L image editing tool during the build process.

Version

Release version of binary image.

ModuleInfoOffset

Offset of first occurrence of AmdModuleHeader relative to the start of the binary image.

EntryPointAddress

Offset of the entry point relative to the start of the binary image.

ImageBase

Image base address. This the linear/physical address to which the image code has presently been located (where it expects to execute).

RelocTableOffset

Offset of relocation table, if applicable. Set to 0 if table not present. This the offset relative to the start of the binary image.

ImageSize

Size, in bytes, of the complete binary image including the header.

Checksum

Checksum of the binary image. The entire binary image sums to 0. Using a word summation of bytes style algorithm.

ImageType

Image type, for example, 1 – B1, 2 – B2.

V_Reserved

Must be 0.

3.4.2 Module Header

A module header exists for each code module included in the binary image. There may be more than one module, in which case the *NextBlock* parameter is used to chain the modules in the binary image.

Prototype

```
typedef struct _AMD_MODULE_HEADER {
      IN    UINT32      ModuleHeaderSignature;
      IN    CHAR8       ModuleIdentifier[8];
      IN    CHAR8       ModuleVersion[12];
      IN    VOID        *ModuleDispatcher;
      IN    struct _AMD_MODULE_HEADER *NextBlock;
} AMD_MODULE_HEADER;
```

Parameters

ModuleHeaderSignature

Constant value equal to the string “$MOD”.

ModuleIdentifier

Published name for this module.

ModuleVersion

Release version of the module in string format. This string reflects the version numbering scheme defined in “2.1 Version Numbering” on page 22.

ModuleDispatcher

Offset of the module dispatcher relative to the start of the binary image.

NextBlock

Offset of the next AmdModuleInfoBlock relative to the start of the binary image. If no more blocks are present, this is set to 0x00000000.

3.5 Integrated Debug Services

Throughout the core files are macros which import debug services. These are the Integrated Debug Services (IDS) macros further defined in “Chapter 19 IDS Configuration Controls” on page 246. Expansion of all IDS macros is controlled by #define compile switches defined for the platform. When disabled, the macros are not expanded during compile and consume no code space.

Chapter 4 Operational Overview

4.1 AGESA™ Core Software

Figure 4.5 is a representation of the time line of a boot sequence depicting the major tasks performed in an average system. (This representation is not intended to be time proportionately accurate.)

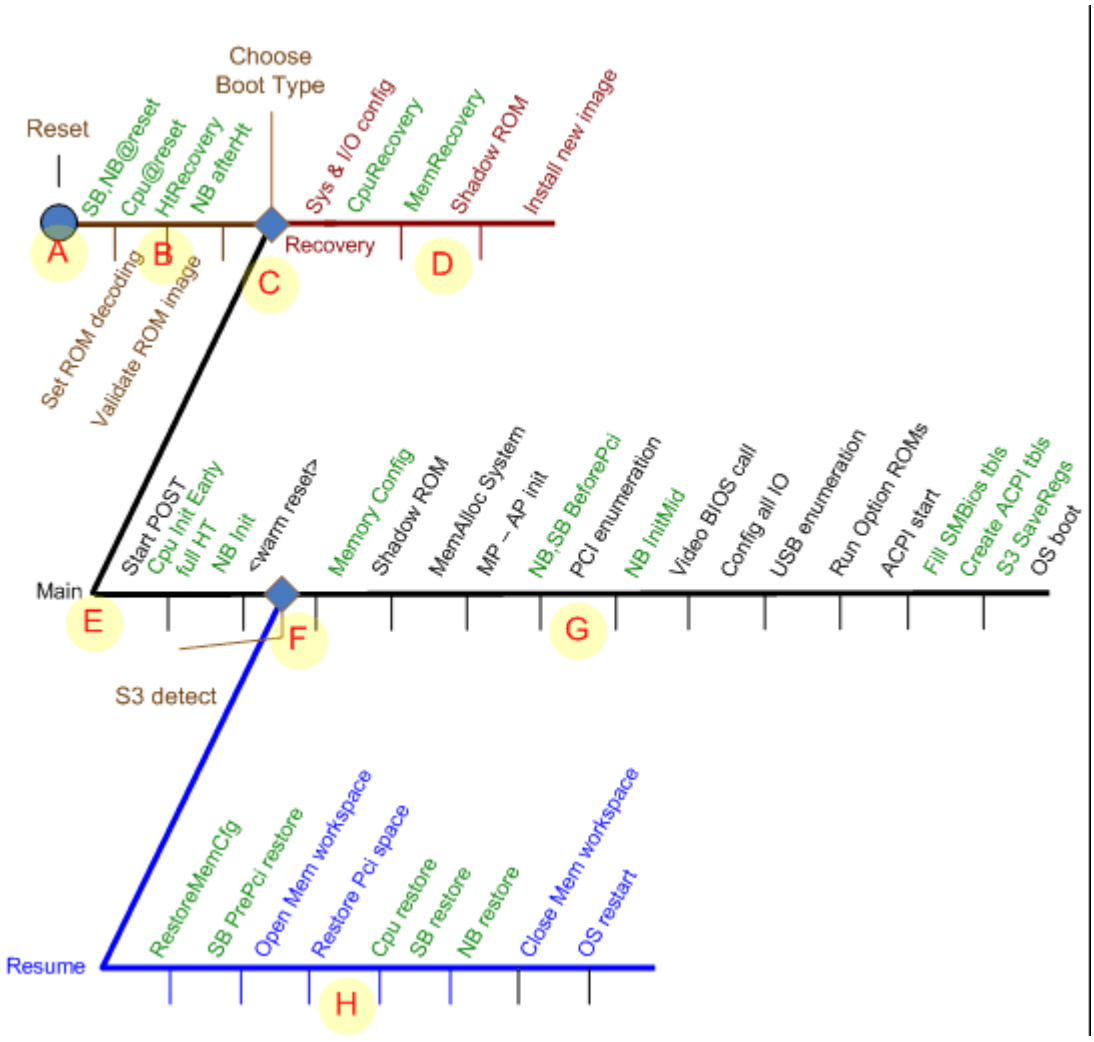

Figure 4.5: General Boot Time Line

Marker points (Figure 4.5):

A — System reset. This occurs at first power-on of the system and also when system resets are performed by the software (warm reset).
B — Immediately after a reset some tasks must be performed in order to make a decision about the integrity of the ROM image.
C — The decision is made about the integrity of the ROM image. If the test fails, execution proceeds to the Recovery Mode. If the test passes, execution jumps to start the main boot sequence.
D — Recovery Mode. The software does minimal hardware initialization, locates the source for a new ROM image and initiates a Flash ROM update.
E — Main boot path. Proceed with full hardware initialization. Warm reset may be needed to instantiate new values into some registers.
F — Determine if the system is resuming from a suspend (ACPI S3) state. If yes, then jump to the restoration sequence.
G — Proceed to OS boot. Initialize all devices, create OS information tables, load the OS from storage, then jump to the OS entry.
H — Restoration. The system and OS state still reside in memory. Restart the system hardware devices, then re-enter the OS.

Many of the activities listed involve AMD silicon devices. The AGESA™ software call entry points are defined to occur at these time points. Figure 4.6 on page 39 shows the approximate positions of the AGESA™ software call entry points.

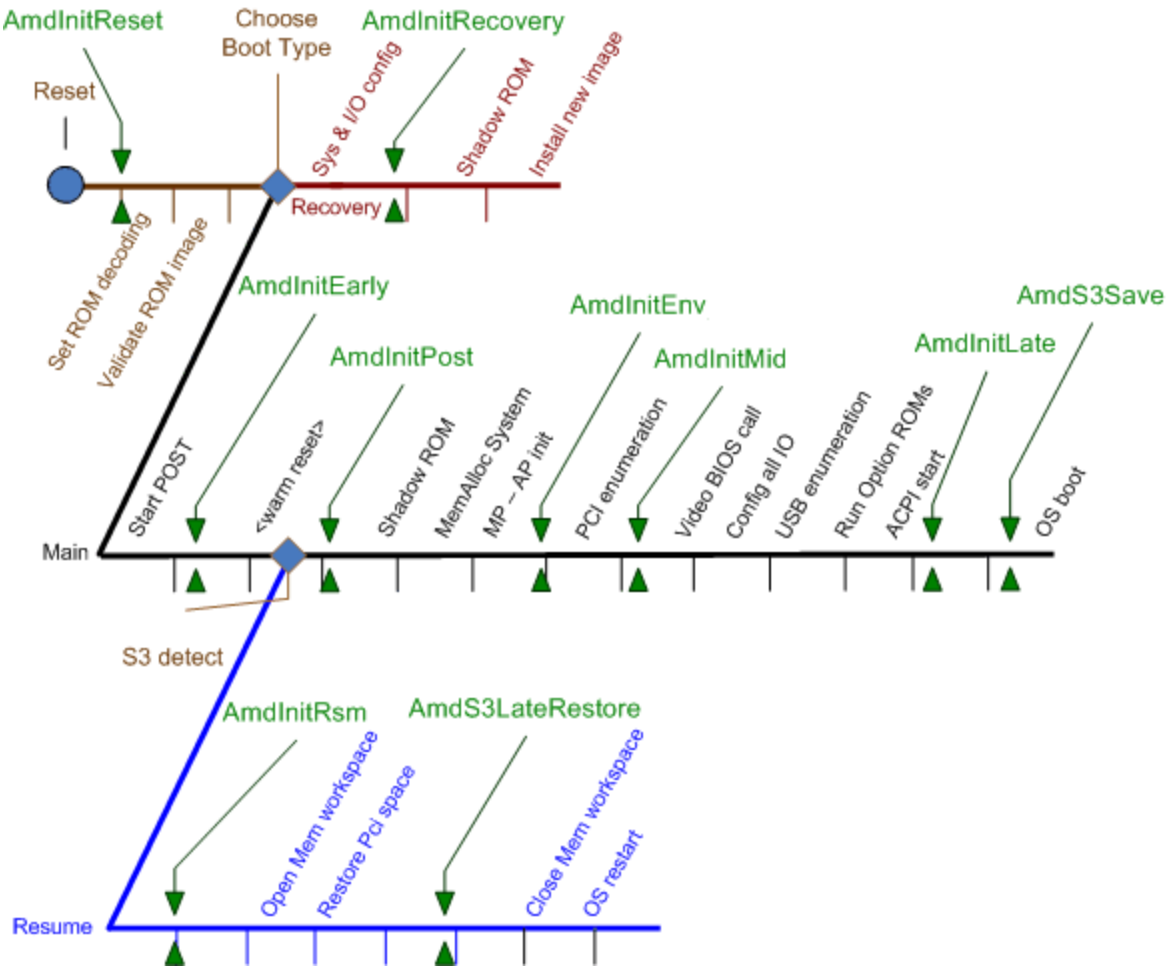

Figure 4.6: AGESA Boot Time Call Points

The call time point duties are outlined as:

AmdInitReset
- initialize heap ctl
- Primary ncHt link initialization
- SB initialization @ reset
- NB after HT

AmdInitEarly
- register load
- full HT initialization
- uCode patch load
- AP launch
- PwrMgmt Init
- NB post initialization
- Detect need for warm reset

AmdInitRecovery <Deprecated>

AmdInitPost
- full memory initialization
- AP data collection (PState leveling)
- HOB xfer from CAR to temp RAM
- AP shutdown
- CAR tear down

AmdInitEnv
- Xfer HOBs from temp to main RAM
- NB before PCI
- SB before PCI

AmdInitMid
- NB GFX bridge initialization

AmdInitLate
- Final CPU restore & configuration
- Fill SMBios/DMI tables
- Create ACPI sub-tables

AmdS3Save AmdInitRtb
- Save proc regs @pointer
- save NB regs @pointer
- save SB regs @pointer

AmdInitResume
- restore memory controller
- exit self refresh
- SB pre-PCI restore

AmdS3LateRestore
- Final CPU restore & configuration
- SB after PCI restore
- NB PCIe S3 initialization
- NB after PCI restore

Details of the actions performed by the AGESA™ software call entry points are listed in Chapter 14:“Entry Point Procedures” on page 98.

4.1.1 Recovery Mode (Deprecated)

The traditional recovery mode operation has been deprecated. Please refer to the UEFI interface descriptions. The ‘miniature’ version of the components used to support the traditional recovery mode are no longer viable and have been replaced with alternative recovery methodologies such as ROM image mirroring.

4.2 Operational Warnings

Do not call functions that are not published in the interface specification. Procedures that are not published are internal functions and may change from release to release.

Do not enable code execution caching before main memory is initialized, except through using the provided AGESA™ software functions. Failure to do so may result in undesired outcomes.

4.3 Reported Errors

The AGESA™ interface calls always return to the host environment. Actions to be taken are indicated by the error class, but the host environment is responsible for implementing the action.

4.3.1 Returned Status Codes

The returned status codes depict the following types of conditions. Please refer to the AGESA.H file for declaration details.

AGESA_SUCCESS No issue. Everything is fine. A log entry may have been made for information purpose. For example, test point, node discovered; ncChain initialized OK; #devices=_N_ (useful for HTX device detection).

AGESA_UNSUPPORTED Used by the optional call-out functions to indicate that the function is not implemented or not supported in the host environment.

AGESA_BOUNDS_CHK Indicates that one of the passed parameters was out of bounds. The output data of the procedure is invalid. Example: the Error Log is empty.

AGESA_ALERT Do not stop POST, but a log event or message was made in which the end user may be interested. The host environment may need to preserve the message and display it or formally log the event. Examples: Sync flood; HT CRC error; MCA status <>0 on reset; thermal event at reset.

AGESA_WARNING Minor degradation in performance or configuration but operation is not prevented. OS boot is permitted. The host environment should notify the user, but may choose to do so only on the first occurrence and not flag every occurrence or on every reset. Examples: Super Weird Mode invoked; HT device lied about ability; unknown CPU revision; Bank interleave not enabled when requested.

AGESA_ERROR Significant degradation of performance, configuration, or operation. The host environment should take corrective actions. OS boot is permitted. The host environment should notify the user of this event every time. Examples: HT fall-back to 1P; HT device initialization failed; Zero common PStates found; Bad DIMM (?).

AGESA_CRITICAL Host environment should stop POST. Attempting to display or log a message is recommended. OS boot should NOT be allowed. Examples: Mixed families found; DP and MP parts mixed; BIST error.

AGESA_FATAL Stop NOW. The host environment must stop POST immediately. Examples: no memory found; processor power > board capacity (possible damage situation).

All reported errors other than SUCCESS and BOUNDS_CHK imply that an event has been logged in the event log.

4.3.2 Error Logging

The event log buffer retains the 16 most recently reported events that may be retrieved using the user event log query interface “AmdReadEventLog” on page 95. The application procedure records various events, including enhanced error descriptions. An detailed explanation of the logged events and errors can be found in Chapter B:“Logged Error Messages” on page 260.

Section II - Porting Guide for Binary Module

Chapter 5 Binary Module Porting

5.1 Overview

All support for 16-bit Real Mode operation has been depricated. The binary module is being retained as it is still viable in the 32-bit environment.

Execution within the binary module occurs in 32-bit protected mode (flat addressing).

5.2 Terms and Definitions

CallOut A CallOut is a call back to the host BIOS made from the AGESA™ software.

Bridge A Bridge is a procedural collection point routine that calls the AGESA™ software binary block from the system BIOS. All of the special details needed to make the call to the binary module are in just one location; not spread throughout the host code.

Dispatcher The Dispatcher is the entry point into the AGESA™ software. It takes a function number as entry parameter in order to invoke the published function.

Router The Router is the CallOut re-entry point to the host environment. It takes a function number as a parameter that selects the call-out function to be executed.

5.3 Theory of Operation

The funneling model, as shown in Figure 5.7, is used to interface the AGESA™ software binary image to a host BIOS. It consists of code compiled into the host environment BIOS, and the AGESA™ software binary image. The host side code (the bridge) calls into the AGESA™ software side (the dispatcher), executes the requested function, and returns to the caller. The model also provides a framework for the AGESA™ software side to call the host BIOS to perform platform specific operations.

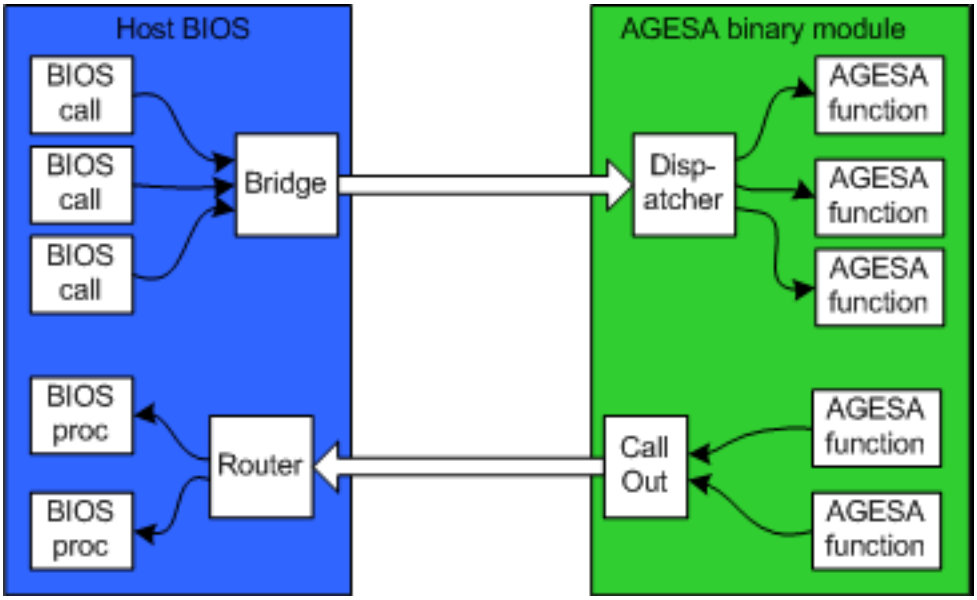

Figure 5.7: Funneling Overview

The bridge code and the BIOS Call router code are built with the host environment BIOS. The dispatcher and call-out port is built with the AGESA™ code into the binary module as described in “Binary Image Design” on page 30.

5.3.1 Calling an AGESA™ Software Procedure

Figure 5.8 displays the steps involved in making a call to the AGESA™ software. This is the typical order followed by the host environment.

At the desired point in the BIOS code, a call is made to the bridge routine with parameters selecting a function. The bridge is the procedure that calls the AGESA™ software dispatcher function in the binary image. The dispatcher calls the desired function, then returns with a status code.

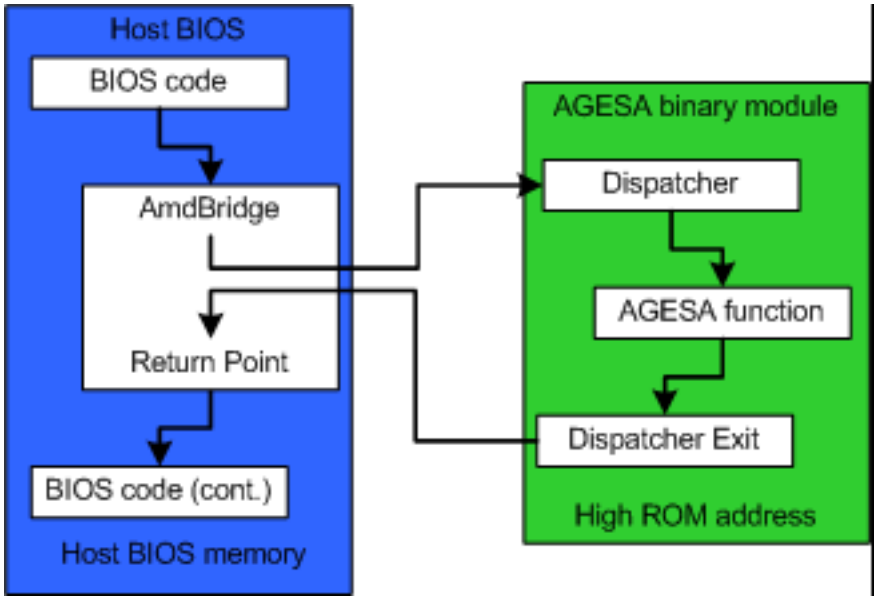

Figure 5.8: Making a ROM image Call

The host BIOS environment bridge routine should treat the call to the dispatcher as a function call with the definition as described in “AmdAgesaDispatcher” on page 48.

5.3.2 Call-Out Operation

Call-out operations work in much the same manner, but in reverse. Figure 5.9 displays the steps involved in making a call out to host environment code from within the AGESA™ software environment.

The AGESA™ software function calls the Call-Out port with parameters selecting the desired host environment procedure. The Call-Out port passes control to the Call-Out router routine that then calls the host environment procedure. When finished, the exit portion of the Call-Out router returns to the calling function.

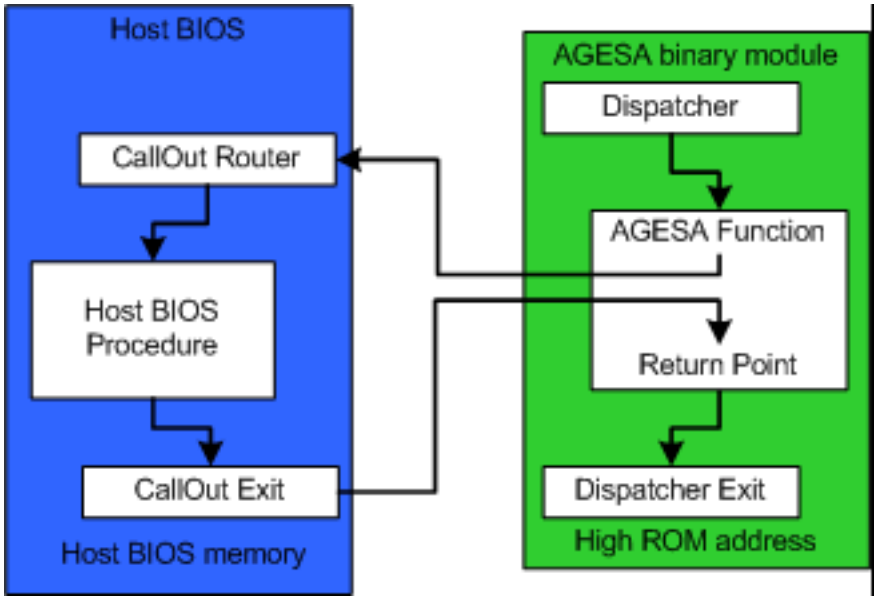

Figure 5.9: Handling a Call-Out

The host BIOS procedure is called from the router as a function call with the definition as described in “Host BIOS Router Procedure” on page 50.

Chapter 6 The Binary Module API

This chapter covers the host environment requirements to implement the host BIOS interface mechanisms to access the AGESA™ software interface.

It is the responsibility of the host environment to establish a valid stack prior to making any call to the binary module interface.

6.1 Making Calls to Core Entry Points

All procedure calls are made using the module dispatchers described in “Binary Image Implementation” on page 34. The host environment procedures must properly create and fill the standard header data.

The identifiers for the published entry point functions can be found in the AMD.H include file.

At each point in the host environment code where a call to the AGESA™ software is desired, a call to an bridge procedure must be inserted. The procedure must locate the binary image entry point as described in“Binary Image Design” on page 30 then make the call to the AMD dispatcher which is defined below. An example of this procedure is available in the file AmdBridge32.asm.

AmdAgesaDispatcher

Execute an AGESA™ software function through the binary module interface. This is the definitioni of the routine contained in the binary module. The host bridge routine must match this definition.

Prototype

```
AGESA_STATUS
AmdAgesaDispatcher (
 IN OUT   VOID *localCfgBlock
)
```

Parameters

localCfgBlock

A pointer to the configuration block (`AMD_CONFIG_PARAMS`). The value of the pointer is determined by either implicit knowledge of the host build environment or by scanning the memory address space for the binary image.

Related Definitions

AMD_CONFIG_PARAMS

The standard header definition shown in “Standard Header” on page 32.

Description

This procedure is the entry gateway or bridge to the binary module code.

An example call sequemce:

```
localCfgBlock.Func = AMD_CREATE_STRUCT;
localCfgBlock.ImageBasePtr = AMD_MODULE_OFFSET;
localCfgBlock.AltImageBasePtr = 0;
localCfgBlock.PcieBasePtr = PCIE_BASE_ADDRESS;
localCfgBlock.CalloutPtr = hostCallOutRouter;
AmdAgesaDispatcher(localCfgBlock);
```

Dependencies

This procedure requires a stack. The host environment must establish the stack environment prior to making the call to this procedure.

6.2 Handling Core Call-Out Procedures

When using the binary image build model, all of the interface call-out functions defined in “Chapter 15 Call-Out Procedures” are collected and funneled through one interface portal, as described by the “Call Out” box in Figure 5.9 on page 46. This provides the host environment with a single point for filtering and dispatching the functions and keeps the call interface consistent over time.

All AGESA core call-outs are made using the call port as illustrated in Figure 5.9 on page 46. This means that the host environment must implement a call-out router function and properly load its address in the standard header. This procedure is implemented in the host environment and functions as described in “Call-Out Operation” on page 45.

The router translates the function number into an execution address for the call-out procedure. The call interface the router procedure exports must match the following.

Host BIOS Router Procedure CallOut

The router function is in the host environment space (external to the AGESA™ software binary module) and routes the AGESA code call-out to a host environment procedure that provides the requested service.

Prototype

```
AGESA_STATUS
CallOutRouter (
     IN UINT32                Function,
     IN UINTN                 FcnData,
     IN CALL_OUT_PARAMS       *ConfigPtr);
```

Parameters

Function *IN*

An index or identifier used to select which call-out function operation is to be invoked. The definition of available functions is available in the AMD.H include file.

FcnData *IN*

A data element specific to the individual call-out functio, meaning or purpose of the data is function-specific. This parameter is passed without conversion or translation, the call-out function may, for example, use this to select a sub-function operation.

ConfigPtr *IN OUT*

Pointer to the configuration data block in use by the calling procedure. This structure includes the standard header structure defined in “Standard Header” on page 32, which contains important information used by the router.

Related Definitions

AMD_CONFIG_PARAMS

The standard header definition shown in “Standard Header” on page 32.

Description

This procedure is called by the Agesa call out port inside the binary module to access services from the host environment. This procedure is implemented in and by the host environment.

Dependencies

The procedure call is made using 32-bit protected mode flat addressing model.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_UNSUPPORTED	The selected function is not implemented by the host environment.

Section III - Porting Guide for UEFI

Chapter 7 UEFI Porting

7.1 Overview

The present UEFI BIOS environment offers four stages of execution (SEC, PEI, DXE, and RTP) spanning three different execution modes, resulting in several pieces of code targeted for the following combinations:

1. 16-bit SEC
2. 32-Bit SEC
3. 32-Bit Platform Initialization (PEI Phase)
4. 64-Bit Platform Initialization (DXE Boot Services Phase)
5. 64-Bit Run Time Phase

To facilitate these combinations, the AGESA™ software core files are arranged into three library modules or groups.

1. AgesaSec (Includes) containing 16-bit execution code
2. AgesaPeiCommonLibs containing 32-bit execution code
3. AgesaDxeCommonLibs containing 64-bit execution code

In addition to these, the UEFI portion of the AGESA™ software adds four more libraries:

4. AgesaPlatformLib containing platform specific materials
5. AgesaPpiLib containing the PEI PPI code files
6. AgesaGuidLib containing GUID information
7. AgesaProtocolLib containing the DXE Protocol files

These libraries are used in the generation of four build targets which will be loaded into the platform UEFI FFS ROM image:

- AmdProcessorInitPeim
- AmdAgesaDxeDriver
- AmdResetManagerPeim
- AgesaProtocolLib

7.2 Tool Set

For the 16-bit SEC phase development, the following tools are used:

- MASM 6.15—To assemble 16-bit code for SEC phase.

For the 32-bit SEC phase and 32-bit platform initialization phase development, the following tools are used:

- ml — Microsoft Macro Assembler Version 8.00.50727.762
- CL — Microsoft 32-bit C/C++ Compiler Version 14.00.50727.42

For the 64-bit platform initialization phase and 64-bit run time phase development, the following tools are used:

- ml — Microsoft Macro Assembler Version 8.0.40310.39
- CL — Microsoft 64 bit C/C++ Compiler Version 14.0.40310.41

7.3 Configuration Options

All the configuration options available for any specific platform build are described in Chapter Chapter 16 on page 176. Refer to the specified section for the details about each configuration option. The options listed here are additional options for the UEFI environment. They are also selectable through the <plat>Option.c file where <plat> is the name of the AMD platform solution set to which the platform belongs.

7.3.1 Build Time Options

There are no additional user options specific to the UEFI environment at this time.

7.3.2 Run Time Options

There are none defined at this time.

7.4 Build Environment Installation

This section talks about items that the action of porting these drivers to the host environment must understand or perform.

- UEFI re-definition of the StdHeader content - The 1st reserved element is used to store the handle for the PEI_Services so that the library level routines can connect with the PeiServices to perform their operations.
- UEFI uses special environment specific library routines as mentioned above to connect to services through the PeiServices PPI.

Chapter 8 The SEC Phase

The purpose of the SEC phase is to establish the C programming language environment - enter 32-bit execution mode and set up a stack region.

The AGESA™ software provides “AMD_CAR_SUPPORT”, documented on page 73, to perform this duty. The host environment must include this procedure.

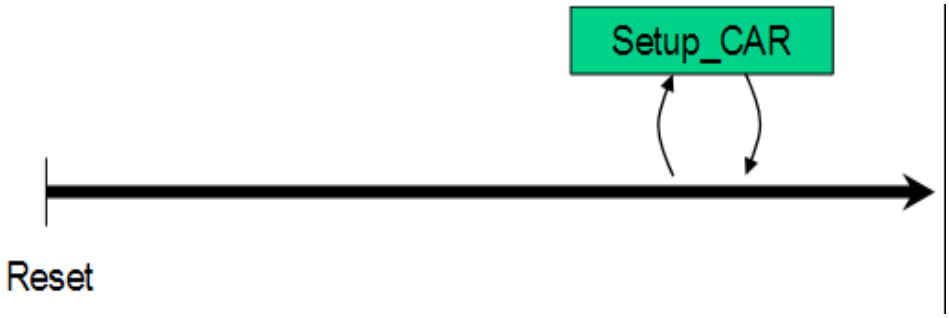

Figure 8.1: Overview of AMD SEC components

Chapter 9 The PEI Drivers

The PEI Driver needed to control the AMD initialization is divided into two PEIM modules, based upon the execution time line. This two PEIMs to support the AGESA™ software are:

- AmdProcessorInitPeim
- AmdInitPostPeim

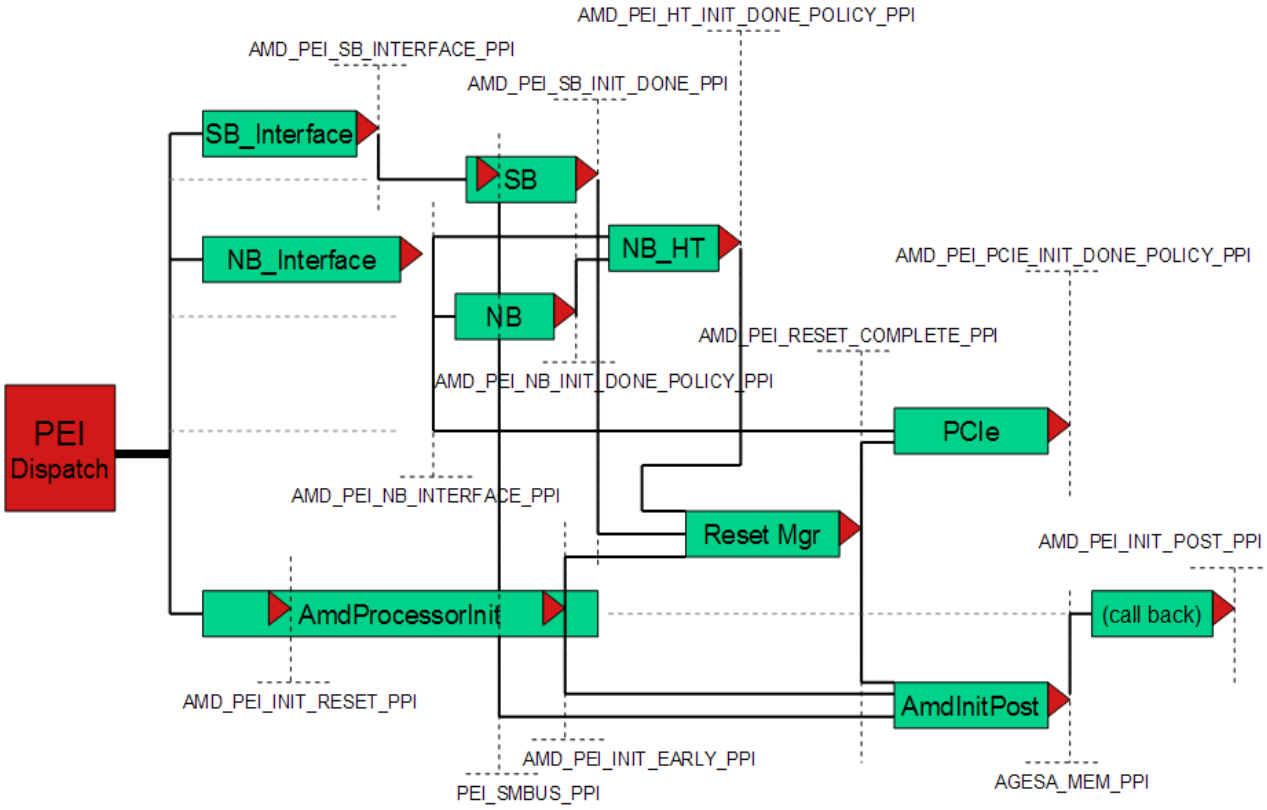

Figure 9.1: Overview of AMD PEI module interactions

9.1 AmdProcessorInitPeim

This PEIM is responsible for initializing all the processor related components and subcomponents. This PEIM incorporates calls to the AGESA™ software core entry points:

- "AmdInitReset", defined on page 99.
- "AmdInitEarly", defined on page 103.

The PEIM will perform processor initialization, then publish the AmdCpuEarlyInit PPI. This allows any component depending upon processor initialization an opportunity to perform any post initialization operations they require.

This PEIM will register for a callback upon publication of the Memory Init PPI when the internal memory initialization completes.

This PEIM consumes the following events:

- MemoryInitPPI

This PEIM produces the following events (PPIs):

- AmdInitResetPPI
- AmdInitEarlyPPI
- AmdInitPostPPI

AmdInitResetPPI

This PPI is published after early processor core initialization is complete.

GUID

```
#define AMD_PEI_INIT_RESET_PPI_GUID \
  { 0xba4ee111, 0xe663, 0x4c4f, 0x89, 0xc5, 0xf3, 0xea, 0x11, 0x64, 0xdd,
0xaf }
```

PPI Interface Structure

```
typedef struct _AMD_PEI_INIT_RESET_PPI {
      UINTN       Revision;
} AMD_PEI_INIT_RESET_PPI;
```

Parameters

Revision

Revision number for this PEIM driver.

AmdInitEarlyPPI

This PPI is published after the call to AmdInitEarly returns and early processor core initialization is complete. This PPI may be used as a callback trigger or dependency for any other module that needs to run right after the processor core initialization completes.

GUID

```
#define AMD_PEI_INIT_EARLY_ PPI_GUID \
      { 0x9cf93fd4, 0xc274, 0x416d, { 0xab, 0x89, 0xe, 0xf0, 0x8f, 0x28,
0x6a, 0xc0 } }
```

PPI Interface Structure

```
typedef struct _AMD_PEI_INIT_EARLY_PPI {
      UINTN       Revision;
} AMD_PEI_EARLY_INIT_PPI;
```

Parameters

Revision

Revision number for this PEIM driver.

AmdInitPostPPI

This PPI is published after the call to AmdInitPost is complete.

GUID

```
#define AMD_PEI_INIT_POST_PPI_GUID \
  { 0xb83e4633, 0xd9a, 0x4463, 0x95, 0x86, 0x62, 0xe7, 0x22, 0xf6, 0xce,
0x48 }
```

PPI Interface Structure

```
typedef struct _AMD PEI_AMD_INIT_POST_PPI {
      UINTN       Revision;
} PEI_AMD_INIT_POST_PPI;
```

Parameters

Revision

Revision number for this PEIM driver.

9.2 AmdInitPostPeim

This PEIM is responsible for initiating the post reset initialization duties.

This PEIM incorporates calls to the AGESA™ software core entry points:

- "AmdInitPost", defined on page 114.
- "AmdInitEnv", defined on page 124.

This will initialize the memory subcomponents and publish the AmdInitPostPPI as indication that memory initialization has succeeded, which helps to launch any other post reset operation by other system modules.

This PEIM consumes or depends upon the following events:

- AmdInitEarlyPPI
- AmdResetComplete

This PEIM publishes the following events (PPIs):

- Memory Init PPI

MemoryInitPPI

This PPI is published to indicate that the main memory initialization has completed. This is used as an internal coordination event to notify the AmdProcessorInitPeim to perform post memory initialization clean-up and configuration. The system memory is not available for general system use until posting of the standard EfiMemory PPI.

GUID

```
#define AMD_MEM_INIT_PPI_GUID = \
      { 0x8ba51c1c, 0x4b30, 0x47df, {0xa4, 0x4e, 0x22, 0x53, 0xe1, 0xa5,
0xd4, 0x2} }
```

PPI Interface Structure

```
typedef struct _AGESA_MEM_PPI {
  UINTN     Revision;
} AGESA_MEM_PPI;
```

Parameters

Revision

Revision number for this DXE driver.

9.3 AmdResetManager

This PEIM coordinates reset requests from multiple AMD PEIMs: the Chipset Northbridge, the SouthBridge along with the processor PEIM. Once loaded, this PEIM will check the status of the PPIs published by the other PEIMs to see is any of them have requested a warm reset to occur. If so, then the reset is generated; if not, then the PEIM will publish the reset-is-complete PPI to flag other PEIMs that they may continue with their post reset duties.

This PEIM consumes or depends upon the following events:

- AmdInitEarlyPPI

This PEIM publishes the following events (PPIs):

- AmdResetComplete

Chapter 10 The DXE Driver

The purpose of the DXE driver is to perform the late initialization and to supply to the system information about the AMD components, in the form of data tables.

The DXE driver incorporates calls to the AGESA™ software core entry points:

- “AmdInitMid”, defined on page 131.
- “AmdInitLate”, defined on page 133.

The AmdInitLate entry point generates several tables used by other drivers in the system. The AMD CPU DXE driver will make the call to AmdInitLate then store the generated tables for later use. A protocol is published, once the tables are available, for the other drivers to use to access the table data.

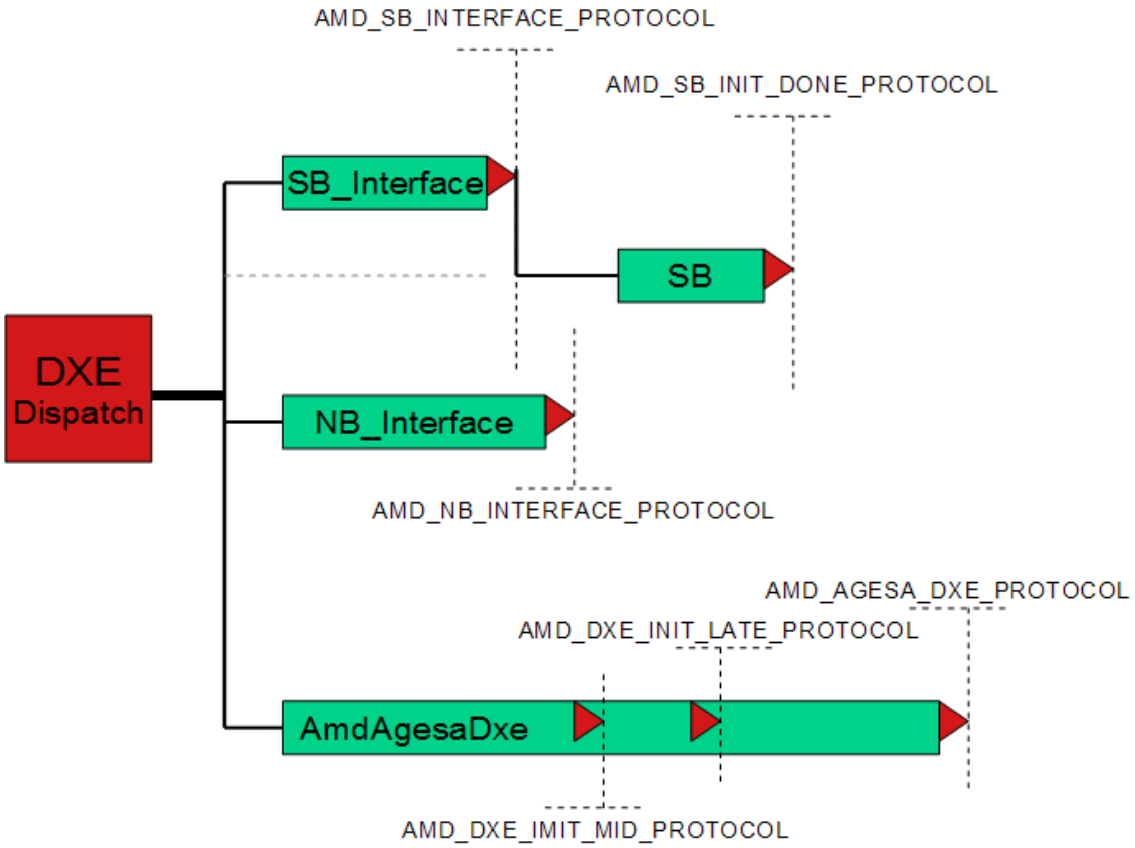

Figure 10.1: Overview of AMD DXE driver interaction

AgesaDxeProtocol

The AgesaDxeProtocol is published as part of AmdCpuDxeDriverInit. This published protocol will be called during update of various tables during POST.

```
#define AGESA_DXE_PROTOCOL_GUID \
  { 0xcdc636f9, 0x55be, 0x4a20, 0x99, 0x2f, 0x38, 0xe8, 0xcc, 0x8f, 0x63,
0x25 }

//
// Protocol prototype
//

typedef enum {
  UndefinedProcessorTableType = 0,
  AcpiPstateType,
  AcpiSratType,
  AcpiSlitType,
  AcpiWheaHestCmcType,
  AcpiWheaHestMceType,
  DmiTableType,
  AcpiAlibType,
  AcpiIvrsType,
  MaxProcessorTableType
} AMD_PROCESSOR_TABLE_TYPE;

typedef EFI_STATUS (EFIAPI *AMD_CREATE_PROCESSOR_TABLES) (
  IN        AGESA_DXE_PROTOCOL            *This,
  IN        AMD_PROCESSOR_TABLE_TYPE      TableType,
  IN OUT    UINTN                         *BufferLength,
  IN OUT    VOID                          **BufferPtr
);

typedef struct _AGESA_DXE_PROTOCOL {
            UINTN                         Revision;
     OUT    AMD_CREATE_PROCESSOR_TABLES   CreateProcessorTables;
  IN OUT    AMD_MID_PARAMS                *MidParamsPtr;
  IN OUT    AMD_LATE_PARAMS               *LateParamsPtr;
} AGESA_DXE_PROTOCOL;
```

AmdGpuVgaControlDxe

The AmdGpuVgaControlDxe driver controls VGA resource decoding for the processor GPU (iGPU).

When installed, the driver will monitor requests through the PciIo protocol and will enable VGA decoding only if the host environment requests this using the attributes of the PciIo protocol for the iGPU.

This monitoring is performed if the driver is installed. There is no further interface to this driver.

Chapter 11 UEFI Porting Check List

11.1 Example Files

The files in the \Addendum directory of the release package are reference files showing examples of how to implement certain functions. The following files are for reference and are not intended to be included in any build.

Cpcar32.asm	This is an example of a wrapper file installing the Cache-As-Ram code into a 32-bit host environment.

The following file is intended for the host environment to copy this file to a platform tip build directory and be modified as needed to describe the specific platform. The settings in the \Addendum copy of this file will reflect the default settings determined by AMD.

<Plat>Options.c	This is the platform description of build options listed in section "Build Customization" on page 69 and "Build Configuration Elements" on page 182. This file is expected to be copied under the platform tip build directory. That directory is also expected to be the target of the vcproj build output files.
<Plat>RecoveryOptions.c	Similar to above, this file is expected to be in the platform target build directory and describes the platform build options to be used for recovery mode.

11.2 Build Control Files

For the UEFI build style, the control files are located in various directories under the \UEFI directory of the release package. These control files need to be integrated into the platform and host environment build systems.

AgesaPei0Libs.inf	This is a control file used to establish the required environment for the AGESA™ Software build process to be used in PEI Phase. The parameters for this console command line program are described in the file header.
AgesaPei1Libs.inf	This is a control file used to establish the required environment to build AGESA PEIM
AgesaDxe0Libs.inf	This is the file used to establish the required environment for AGESA™ software build process to be used in AGESA DXE Drivers
AgesaDxe0Libs.inf	This is the file used to establish the required environment to build AGESA DXE Drivers.

11.3 Environment Variables

The AGESA™ software build environment requires a few environment variables to be set in order to function properly. For the UEFI build, these variables are set in the platform master build control file and are expected to be set by the host environment prior to invoking the AGESA™ software build control .inf files.

- AGESA_ROOT Identifies the directory where the AGESA™ code is located. The top most directory, where AGESA.h is located. This must be the full path (e.g. D:\myPlatform\AGESA\ ) and includes the trailing '\'.
- AGESA_OptsDir Identifies the platform tip build directory where the platform options (<Plat>Options.c) and customization files are located. This must be the full path (e.g. D:\myPlatform\MotherBoards\Project\AGESA\ ) and includes the trailing '\'.
- Solution Identifies the name of the AMD Platform Solution being targeted for the build (e.g. "Maranello").

11.4 UEFI Check List

Points to remember:

- Copy the <plat>Options.c file from the \Addendum directory to the platform tip build directory. AMD recommends the use of a sub-directory named 'AGESA' to contain these files and the build output files.
- Copy the OptionsIds.h content in "Example" on page 254 to OptionsIds.h in the platform build tip directory and make changes to enable the IDS support desired. *It is highly recommended that you set the following for initial integration and development:*

```
#define IDSOPT_IDS_ENABLED          TRUE
#define IDSOPT_ERROR_TRAP_ENABLED   TRUE
#define IDSOPT_ASSERT_ENABLED       TRUE
```

- Edit and modify the option selections in those two files to meet the needs of the specific platform.
- Set the environment variable 'AGESA_ROOT' as noted above.
- Set the environment variable 'AGESA_OptsDir' as noted above.
- Generate the doxygen documentation as described in "Internal Documentation" on page 25, or locate the file *arch2008.chm* within your AGESA™ release package.
- Enable CAR (Cache As Ram) before the call to AmdInitReset by using "AMD_CAR_SUPPORT" on page 73.
- Use the 'ByHost' allocation method for the call to AmdInitReset. Allocate the structure space before the call.
- The host environment code must not modify the SS or ESP registers while the CAR is enabled.
- The host environment MUST NOT modify the cache controls (MTRRs, etc) during the Cache-As-Ram time period (reset through final return from AmdInitPost).

Section IV - API Specification

Chapter 12 Introduction

This chapter describes some basics related to building or compiling the core code files.

The features that are available are arranged into sets for the various platform solution product lines. So the first level of selection for the user is to specify the platform solution to which their platform belongs. The second level of selection is the build method, then the third level is the individual options.

12.1 Platform solutions

A “platform solution” is a collection of AMD products that are related or are compatible for constructing a platform. A code name is assigned to each “solution” as a marketing reference. Current examples of platform solution names are “Danube” and “Maranello”. For the rest of this chapter the generic indicator of <plat> is used.

12.2 Build Methods

There are two primary methods to build the AGESA™ software core files as listed below. A project control file is provided for each method.

Binary Modules

AMD distributes the source code and build control files to create binary modules as described in “Binary Image Design” on page 30. This is referred to as the indirect method, since the interface calls are routed through a common entry point.

Source Integration

In the source integration method, the host environment integrates the distributed source files into their environment and builds a ROM image using their tool set. This is also referred to as the integrated or direct interface.

12.3 Build Customization

Build time defaults may be customized easily using the <plat>Options.c file for each platform. The full set of build customizations is presented in “Build Configuration Elements” on page 182. Because

appropriate defaults are provided automatically, it is only necessary to specify build customizations for items that need to be changed for the platform.

Chapter 13 Service Procedures

13.1 Stack Procedures

These stack procedures are provided to the host environment for use in establishing the stack environment. Since these procedures must be integrated into the host environment, they are provided as included code in the form of a macro. These must be called by the host environment.

The stack routines do not rely on segment names being imported from the host environment through include files. The procedures defined below are without segment declarations. The host environment must define a local file that establishes the proper execution environment for their needs and use the INCLUDE directive to pull in the stack code source file into the local file.

The routines are native assembly code. Their purpose is to establish or destroy the stack and therefore expect to execute in a stackless mode. They can be included into either an assembly host environment local file or, with care, can be included into a host environment C language file.

AMD_RESET_ENABLEMENT

This macro contains AMD silicon initialization controls that are required extremely early in POST.

Prototype

```
OEM_SEGMENT1_START
;AMD_RESET_ENABLEMENT; Host Environment Procedure
; Input:    EBX   - Return address
PlatformReset PROC FAR PUBLIC

     AMD_RESET_ENABLEMENT
     jmp     ebx               ;Return to caller
PlatformReset ENDP

OEM_SEGMENT1_END
```

Parameters

none

Description

In the UEFI implementations, the earliest control point for AMD software is in the PEI phase. This may be too late for some critical settings. This macro provides a very early control point for the AMD software to establish the critical settings.

This macro should be placed as early as possible in the boot sequence. It must be before the use of the AMD_ENABLE_STACK macros and should be before entry to 32bit execution mode.

Dependencies

Registers destroyed: MMX0, all general purpose registers (except EBX).

Specifically preserved is the EBX register - for use as a stackless return.

Status Codes Returned

None.

AMD_CAR_SUPPORT

This macro is used to reduce the code size for establishing the Cache-As-RAM (CAR) support in early POST. This optional macro will select the specific SoC for which the BIOS is being built.

Prototype

```
AMD_CAR_SUPPORT <SolutionName>
```

Example:

```
OEM_SEGMENT1_START
;EnableStack      ; Host Environment Procedure
; Input:    EBX   - Return address
EnableStack PROC FAR PUBLIC
     AMD_CAR_SUPPORT Mullins

     AMD_ENABLE_STACK
     jmp     ebx               ;Return to caller
EnableStack ENDP

OEM_SEGMENT1_END
```

Parameters

SolutionName IN

The name of the SoC or ‘solution’ being built. The current members in this list are: ‘Kaveri’, ‘ Richland’, ‘Kabini’, ‘Kyoto’, ‘Mullins’.

Description

This macro will select the target SoC for inclusion in the build. All other processor families will be excluded. This macro must be placed before the use of the AMD_ENABLE_STACK macros. If it is not present, CAR code for all supported families will be included.

If you need to specify more than one SoC, use the macro once for each SoC name. Example:

```
AMD_CAR_SUPPORT Kabini
AMD_CAR_SUPPORT Mullins
AMD_CAR_SUPPORT Kaveri
```

Status Codes Returned

An error will be generated during compilation if the SoC SolutionName is not recognized.

AMD_ENABLE_STACK (Deprecated)

AMD_ENABLE_STACK_AT_BOTTOM (Deprecated)
AMD_ENABLE_STACK_AT_TOP (Deprecated)

AMD_DISABLE_STACK (Deprecated)

AMD_ENABLE_UEFI_STACK

This procedure is used to establish the stack within the host environment. Two variations are available for controlling stack placement:

AMD_ENABLE_UEFI_STACK_AT_BOTTOM
AMD_ENABLE_UEFI_STACK_AT_TOP

AMD_ENABLE_UEFI_STACK is equivalent to AMD_ENABLE_UEFI_STACK_AT_BOTTOM.

Prototype

```
OEM_SEGMENT1_START
;EnableStack      ; Host Environment Procedure
; Input:    EBX   - Return address
EnableStack PROC FAR PUBLIC
     AMD_ENABLE_STACK
     jmp    ebx               ;Return to caller
EnableStack ENDP

OEM_SEGMENT1_END
```

Parameters

EAX *IN*

The BIST value to save.

ECX *IN*

Size in bytes of initial execution cache allocation. Execution cache allocation will be made immediately below the 4 GByte boundary.

SS:ESP *OUT*

Points to the private stack location for this processor core.

EAX *OUT*

Contains the AGESA_STATUS return code.

EDX *OUT*

Contains the event sub-class for status return codes of higher severity than AGESA_SUCCESS.

ECX *OUT*

Upon success, contains this processor core's stack size in byte.

EDI *OUT*

Points to the stack frame. The stack frame will be initialized as follows below.

```
[EDI]UEFI_SEC_PEI_HAND_OFF.BOOT_FIRMWARE_VOLUME_BASE = OEM_BFV_BASE
[EDI]UEFI_SEC_PEI_HAND_OFF.BOOT_FIRMWARE_VOLUME_SIZE = OEM_BFV_SIZE
```

```
[EDI+sizeof(UEFI_SEC_PEI_HAND_OFF)].OEM_DATA_DWORD[0] = BIST
```

Related Definitions

The EBX register is fully preserved by this routine. This provides the possibility for the host environment to use the EBX register as a return address at which execution resumes following the completion of the procedure.

The procedure will use the build time value of AMD_CAR_STACK_FRAME_PAD to determine the size of the SEC to PEI handoff stack frame. The default pad value is zero, which allocates four bytes for saving the BIST value.

Description

This procedure is native assembler source file and is “stackless” in its operation. Therefore the interface uses a register passing model.

The purpose is to initialize the processor and cache system to establish a viable stack region prior to main memory being available. The available pre-memory stack region is divided among the processor cores according to application needs. It then maps them to global address space and sets the processor MTRRs accordingly. This routine must be run on all processor cores.

The procedure checks whether protected mode is enabled and if so, does not modify the Stack Segment register (SS). In this case the host environment must point SS to a GDT descriptor with base address = 0x0000_0000 before entering this procedure. The Stack Pointer (ESP) is set to contain a 32-bit memory offset. If Real Mode is being used, SS:ESP is set to point to a 16-bit execution-compatible Segment:Offset format. The upper 16 bits of ESP will be zeroes.

The procedure will create an instance of UEFI_SEC_PIE_HAND_OFF in the specified stack frame area.

Dependencies

The host environment must use this procedure and not rely on any other sources to create the stack region.

Registers destroyed: EAX, ECX, EDX, EDI, ESI

If 16-bit environment, also destroyed: DS, ES

Status Codes Returned

AGESA_SUCCESS	The stack space has been allocated for this core.
AGESA_WARNING	CPU_EVENT_STACK_REENTRY - A stack is in use from a previous invocation of AMD_ENABLE_STACK. The stack will be reset to empty.
AGESA_FATAL	CPU_EVENT_UNKNOWN_PROCESSOR_FAMILY - The stack cannot be created because the processor family is unknown.

AMD_DISABLE_UEFI_STACK

This procedure is used to remove the pre-memory stack from within the host environment.

Prototype

```
OEM_SEGMENT2_START
;DisableStack     ; Host Environment Procedure
; Input:    EBX   - Return address
DisableStack PROC FAR PUBLIC
     AMD_DISABLE_UEFI_STACK
     jmp     ebx
DisableStack ENDP

OEM_SEGMENT2__END
```

Parameters

EAX *OUT*

Contains the AGESA_STATUS return code

Related Definitions

The EBX register is fully preserved by this routine. This provides the possibility for the host environment to use the EBX register as a return address to which execution resumes following the completion of the procedure.

Description

This procedure is native assembler source file and is “stackless” in its operation. Therefore the interface uses a register passing model.

Their purpose is to return the processor and cache system to a non-stack-enabled state. This procedure is expected to be executed only once at the point where main memory becomes available. It is expected that the UEFI PEI core has relocated the stack to memory and the MTRR map has been synchronized. Therefore, this routine will not modify the MCRR settings, but will just disable CAR mode. This procedure must be run on all processor cores. All cache lines are flushed.

The exit state for the BSP is described as follows:

- Processor Cache is enabled (CD bit is cleared).
- MTRRs used for execution cache are kept.
- Cache content is flushed (invalidated without write-back).
- Any family-specific clean-up done.

Dependencies

This procedure should only be invoked by the BSP. The host environment must use this procedure and not rely on any other sources to break down the stack region.

If executing in 16-bit code, the host environment must establish the “Big Real” mode of 32-bit addressing of data.

Registers destroyed: EAX, ECX, EDX, ESI, ESP, EDI

Status Codes Returned

AGESA_SUCCESS	The stack space has been disabled for this core.

13.2 General Service Procedures

These are procedures provided to the host environment as an aid in performing its tasks.

AmdCreateStruct

Creates a storage space for a parameter block of an AGESA™ software call entry.

Prototype

```
AGESA_STATUS
AmdCreateStruct (
     IN OUT  AMD_INTERFACE_PARAMS    *InterfaceParams
);
```

Parameters

InterfaceParams

Pointer to the parameter structure containing the descriptor of the structure to be created.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS       StdHeader;
     IN      AGESA_STRUCT_NAME       AgesaFunctionName;
     IN      ALLOCATION_METHOD       AllocationMethod;
     IN OUT  UINT32                  NewStructSize;
     IN OUT  VOID                    *NewStructPtr;
} AMD_INTERFACE_PARAMS;
```

This is the parameter containment structure for the AmdCreateStruct function. Along with the standard header defined in “Standard Header” on page 32, it contains:

AgesaFunctionName

Name or identifier of the structure to be created. This parameter must be filled by the caller.

AllocationMethod

Identifier for the method to use for allocating the structure storage space. This parameter must be filled by the caller.

NewStructSize

This is the size in bytes of the storage space. Upon return, this value indicates the size of the structure created. If the allocation method specified is “ByHost”, then this parameter must be filled by the caller to indicate the amount of storage the host has already allocated.

NewStructPtr

This is a pointer to the created structure. This value is set by the procedure for use by the caller. The content of the StdHeader is copied to the new structure.

```
typedef enum {
     AMD_INIT_RECOVERY,
     AMD_CREATE_STRUCT,
     AMD_INIT_EARLY,
     AMD_INIT_ENV,
     AMD_INIT_LATE,
     AMD_INIT_MID,
     AMD_INIT_POST,
     AMD_INIT_RESET,
     AMD_INIT_RESUME,
     AMD_RELEASE_STRUCT,
     AMD_S3LATE_RESTORE,
     AMD_S3_SAVE,
     AMD_GET_APIC_ID,
     AMD_GET_PCI_ADDRESS,
     AMD_IDENTIFY_CORE,
     AMD_READ_EVENT_LOG,
     AMD_GET_EXECACHE_SIZE,
     AMD_LATE_RUN_AP_TASK,
     AMD_IDENTIFY_DIMMS
} AGESA_STRUCT_NAME;
```

This is an enumerated list of structure names that can be created.

```
typedef enum {
     PreMemHeap,
     PostMemDram,
     ByHost
} ALLOCATION_METHOD;
```

This is an enumerated list of methods by which to allocate the storage space for the structure.

PreMemHeap	Use this allocation method for calls that occur before the main memory is available. The AGESA™ software uses its internal heap sub-system to allocate space in the Cache-as-RAM storage area.
PostMemDram	Use this allocation method for calls that occur after main memory is available. The call-out “AgesaAllocateBuffer” on page 147 is used to allocate storage space.
ByHost	This allocation method is provided for the host environment to pre-allocate the structure storage space by their own means. The procedure does not attempt to allocate any storage space but still calls the structure initializer functions.

Caution: When using this method, the host environment takes the responsibility to assure and maintain the size of space allocated is sufficient for the target structure.

Note: When using this method, the StdHeader parameter value is used as the base of the storage area. This means that upon a successful completion of the initializer, the target structure definition overlays the initial content of the `CREATE_STRUCT_PARAMS` parameter structure. The AllocMethod, StructSize, and NewStruct values are no longer available. If the pre-allocated space is not sufficient for the target structure an error is returned and the `CREATE_STRUCT_PARAMS` content is retained.

Description

This procedure creates storage space for the indicated structure, then pre-initializes the structure with default values. The storage for the specific structure is allocated per the indicated method, then the initializer function for that structure is called.

Dependencies

The creation and removal of the structure storage depends upon the host environment calling procedure using the AmdCreateStruct and AmdReleaseStruct procedures. Failure to release a structure can cause undesired outcomes.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The target structure has been initialized successfully.
AGESA_ERROR	The storage space for the target structure is insufficient or could not be allocated.
AGESA_FATAL	Unable to initialize structure storage due to unrecognized processor family.

AmdGet2DDataEye

Obtain the data eye results for the specified DIMM.

Prototype

```
AGESA_STATUS
AmdGet2DDataEye (
     IN OUT       AMD_GET_DATAEYE   *AmdGetDataEye
);
```

Parameters

AmdGetDataEye

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS StdHeader;
     IN      AMD_POST_PARAMS   PostParamsPtr;
     IN      UINT8             Socket;
     IN      UINT8             MemChannelId;
     IN      UINT8             DimmId;
     IN      UINT8             DataEyeType;
      OUT    UINT8             *DataEyeBuffer;
} AMD_GET_DATAEYE;
```

Socket

Number of the socket on which the desired DIMM is populated.

MemChannelId

The memory channel on which the desired DIMM is populated.

DimmId

The DIMM for which results are desired.

DataEyeType

Indicates whether to return the data eye results for reads or writes.

DataEyeBuffer

Provides the data eye results as a bitmap representing a composite data eye of all the DIMM byte lanes.

Description

Return the data eye bitmap for the DIMM requested. The bitmap is thirty one elements of thirty two bits each.
Data eye results are normalized relative to the 1D trained Center[+16, -15].

Dependencies

The coherent HyperTransport™ links must be operational so that all processors are available.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

This service call must be made from the AmdInitPost wrapper, before memory structures are deallocated. (See “AmdInitPost” on page 114.)

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AmdGetApicId

Obtain the APIC address for the specified processor core.

Prototype

```
AGESA_STATUS
AmdGetApicId (
     IN OUT       AMD_APIC_PARAMS   *AmdParamApic
);
```

Parameters

AmdParamApic

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS StdHeader;
     IN      UINT8             Socket;
     IN      UINT8             Core;
      OUT    BOOLEAN           IsPresent;
      OUT    UINT8             ApicAddress;
} AMD_APIC_PARAMS;
```

Socket

Number of the socket for which to determine the APIC address.

Core

Number of the processor core relative to the socket, for which to determine the APIC address.

IsPresent

Indicates whether the indicated Socket-Core is present in the system.

ApicAddress

The procedure stores the APIC address for the specified socket and core into this structure element.

Description

Determine the APIC ID or address of the specified processor core. Core numbers are relative to the socket regardless if the processor has multiple die or not.

Dependencies

The coherent HyperTransport™ links must be operational so that all processors are available.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AmdGetAvailableExeCacheSize

Returns available cache size for execution caching in the pre-memory time frame.

Prototype

```
AGESA_STATUS
AmdGetAvailableExeCacheSize (
     IN OUT  AMD_GET_EXE_SIZE_PARAMS *AmdGetExeSizeParams
);
```

Parameters

AmdGetExeSizeParams

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     OUT     UINT32                  AvailableExeCacheSize;
} AMD_GET_EXE_SIZE_PARAMS;
```

AvailableExeCacheSize

The number of bytes available for use as execution cache on the current core.

Description

During the pre-memory time frame, the amount of cache available for execution caching is limited and can vary depending on the processor(s) installed in the system. This routine returns the amount of cache available for use as code execution cache. The value pertains to the current execution core. The customer can use this information to decide the best regions to target for code execution cache. This routine can be executed on all CPU cores. This routine applies to the pre-memory time period only. After main system memory is available, the aforementioned limits are removed.

Dependencies

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AmdGetPciAddress

Obtain the base PCI address for the specified processor die.

Prototype

```
AGESA_STATUS
AmdGetPciAddress (
     IN OUT  AMD_GET_PCI_PARAMS      *AmdParamGetPci
);
```

Parameters

AmdParamGetPci

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS StdHeader;
     IN      UINT8             Socket;
     IN      UINT8             Module;
      OUT    BOOLEAN           IsPresent;
     OUT     PCI_ADDR          PciAddress;
} AMD_GET_PCI_PARAMS;
```

Socket

Number of the socket for which to determine the PCI address.

Module

Number of the die relative to the socket, for which to determine the PCI address.

IsPresent

Indicates if the indicated Socket-Module is present in the system.

PciAddress

The procedure stores the PCI address for the indicated processor module into this structure element. The PCI address includes the segment, bus, and device values. The function and register fields are set to zero.

Description

Determine the base PCI address for the specified processor die. Each die within a processor has a separate PCI device number.

Dependencies

The coherent HyperTransport™ links must be operational so that all processors are available.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AmdIdentifyCore

Identifies the current executing processor core.

Prototype

```
AGESA_STATUS
AmdIdentifyCore (
     IN OUT       AMD_IDENTIFY_PARAMS     *AmdParamIdentify
);
```

Parameters

AmdParamIdentify

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS StdHeader;
     OUT     UINT8             Socket;
     OUT     UINT8             Module;
     OUT     UINT8             Core;
} AMD_IDENTIFY_PARAMS;
```

Socket

The procedure sets this to the number of the socket containing this processor core.

Module

The procedure sets this to the number of the die relative to the socket on which this core resides.

Core

The procedure sets this to the number of this processor core relative to the socket.

Description

This procedure is used to identify the current processor core. The system address is returned as socket, die, and core. This information can be used to determine other system addresses, for example, PCI base address and APIC address.

Dependencies

The coherent HyperTransport™ links must be operational so that this processor available.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AmdIdentifyDimm

Identifies the current executing processor core.

Prototype

```
AGESA_STATUS
AmdIdentifyDimm (
     IN OUT       AMD_IDENTIFY_DIMM       *AmdDimmIdentify
);
```

Parameters

AmdDimmIdentify

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS StdHeader;
     IN      UINT64            MemoryAddress;
     OUT     UINT8             SocketId;
     OUT     UINT8             MemChannelId;
     OUT     UINT8             DimmId;
     OUT     UINT8             ChipSelect;
     OUT     UINT8             Bank;
     OUT     UINT32            Row;
     OUT     UINT16            Column;
} AMD_IDENTIFY_DIMM
```

MemoryAddress

This is the system address which is the target of the inquiry.

SocketId

The procedure sets this to the number of the socket containing the memory responding to the specified address.

MemChannelId

The procedure sets this to the number of the channel relative to the socket that contains the memory responding to the specified address.

DimmId

The procedure sets this to the number of the DIMM relative to the channel that contains the memory responding to the specified address.

ChipSelect

The procedure sets this to the number of the chip select relative to the DIMM that contains the memory responding to the specified address.

Bank

The procedure sets this to the number of the bank relative to the chip select that contains the memory responding to the specified address.

Row

The procedure sets this to the number of the row relative to the bank that contains the memory responding to the specified address.

Column

The procedure sets this to the number of the column relative to the bank that contains the memory responding to the specified address.

The SocketId, MemChannelId, and DimmId values are the same as those provided to the call-out "AgesaReadSpd" on page 155. The DIMM number is directly related to the chip-select control line emanating from the memory controller, as shown in Table 1 on page 156. If the DimmId is greater than the number of DIMMs per channel for that processor, then the DIMM is a quad rank DIMM and DimmId minus two should be used to identify the DIMM.

Description

This procedure is used to identify the Field Replaceable Unit (FRU) memory device (DIMM) that corresponds to the given memory address. This can be useful to locate the FRU that is responsible for a memory failure. The SocketId, MemChannelId, and DimmId parameters provide FRU identification.

This procedure also provides error location information within the FRU. This can be used to support additional error reporting, handling, or testing. For example, it is useful with error injection testing to be able to confirm the error location to the chip select, bank, row, and column location.

Dependencies

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_BOUNDS_CHK	The provided system address is outside the range for the main memory.

AmdAddMmioMapping

User interface call to merge the requested MMIO range with the current processor MMIO address maps.

Prototype

```
AGESA_STATUS
AmdAddMmioMapping (
     IN      AMD_ADD_MMIO_PARAMS     AmdAddMmioParams
);
```

Parameters

AmdAddMmioParams

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS       StdHeader;
     IN      UINT64                  BaseAddress;
     IN      UINT64                  Length;
     IN      PCI_ADDR                TargetAddress;
     IN      AMD_MMIO_ATTRIBUTE      Attributes;
} AMD_ADD_MMIO_PARAMS;
```

StdHeader

AMD standard Header structure. See “Standard Header” on page 32.

BaseAddress

This is the starting address of the requested MMIO range.

Length

This is the length of the range to allocate, in bytes.

TargetAddress

This is the PCIe® address of the device for which this range is allocated, and it provides the bus, device, and function of the target device.

Attributes

This indicates the attributes of the requested range.

```
typedef struct {
     UINT8   MmioReadableRange:1;
     UINT8   MmioWriteableRange:1;
     UINT8   MmioPostedRange:1;
     UINT8   MmioSecuredRange:1;
     UINT8   :3;
     UINT8   OverrideExisting:1;
```

```
} AMD_MMIO_ATTRIBUTE;
```

MmioReadableRange

The requested range supports reads.

MmioWriteableRange

The requested range supports writes.

MmioPostedRange

The requested range supports posted transactions.

MmioSecuredRange

The requested range is secured.

OverrideExisting

If this request overlaps an existing allocation which has different attributes, this request should override the attributes for that overlapping range. Otherwise, the recommended and default behavior is to consider conflicting attributes as an error.

Description

User interface call to update the MMIO address mappings to include a requested range.

Dependencies

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_BOUNDS_CHK	One or more input parameters are invalid. For example, the TargetAddress does not correspond to any device in the system.
AGESA_ERROR	The requested range could not be added because there are not enough mapping resources.

AmdReadEventLog

User interface call to obtain the next event log entry.

Prototype

```
AGESA_STATUS
AmdReadEventLog (
     IN      EVENT_PARAMS      *Event
);
```

Parameters

Event

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS StdHeader;
     OUT     UINT32            EventClass;
     OUT     UINT32            EventInfo;
     OUT     UINT32            DataParam1;
     OUT     UINT32            DataParam2;
     OUT     UINT32            DataParam3;
     OUT     UINT32            DataParam4;
} EVENT_PARAMS;
```

StdHeader

AMD standard Header structure. See “Standard Header” on page 32.

EventClass

This is the AGESA_STATUS value reported to the caller by the application. This should match the error returned so the caller can match up the log entry with the returned error status.

EventInfo

This is a unique identifier to specify the application. It has bit fields defined by the application that are unique. A returned value of zero indicates the log was empty.

DataParam[1:4]:

These are 4 data values provided by the application for logging. The meaning of the data is application-specific. Unused parameters are padded with 0.

Further information about the logged events can be found in “Logged Error Messages” on page 260.

Description

User interface call to obtain the next event log entry. User should note that there may be more than one log entry created by an application and should not assume only the first entry relates to the error returned by the application. The AmdReadEventLog routine reads the oldest entry from the circular buffer and places that information to the structure pointed to by the parameter. The internal pointers are incremented to remove the entry from buffer so that a subsequent call returns the next entry from the buffer. If the buffer is empty, the EventInfo value will be zero.

Dependencies

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AmdReleaseStruct

Clears a storage space from allocation for a parameter block of an AGESA™ software call entry.

Prototype

```
AGESA_STATUS
AmdReleaseStruct (
     IN OUT  AMD_INTERFACE_PARAMS    *InterfaceParams
);
```

Parameters

InterfaceParams

Pointer to the parameter structure that contains the descriptor information about the structure to be released.

Related Definitions

`AMD_INTERFACE_PARAMS`

This structure is the same as was defined in "AmdCreateStruct" on page 79.

Description

This procedure releases the storage space previously allocated for the interface structure.

This procedure may invoke calls to the following call-outs:

- "AgesaDeallocateBuffer" on page 149.

Dependencies

The creation and removal of the structure storage depends upon the calling procedure using the AmdCreateStruct and AmdReleaseStruct procedures. Failure to release a structure can cause undesired outcomes.

Status Codes Returned

AGESA_SUCCESS	The storage space was released successfully.

Chapter 14 Entry Point Procedures

Entry procedures are listed in time point order.

Warning: Only the published entry functions are assured to be available over time. Use of other procedure names by the host environment that are found by source code inspection is strongly discouraged. AMD reserves the right to modify or remove internal procedure names not published in this specification.

14.1 Recovery Branch Functions

These functions should be located in the protected “boot block” section of the flash ROM. They set the base functionality in preparation to execute a flash ROM update utility. Performance and capability are not optimized. These routines are designed to establish operating settings applicable for present a some future upgrades to the target processor. Because today’s code cannot possibly know about tomorrow’s upgrades, only a common minimum set of capability is attempted.

AmdInitRecovery (Deprecated)

AmdInitReset

This procedure performs minimal basic processor initialization after a system reset.

Prototype

```
AGESA_STATUS
AmdInitReset (
     IN OUT  AMD_RESET_PARAMS  *ResetParams
);
```

Parameters

ResetParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS       StdHeader;
     IN      EXECUTION_CACHE_REGION  CacheRegion[3];
     IN      FCH_RESET_INTERFACE     FchInterface;
} AMD_RESET_PARAMS;
```

CacheRegion

This is an array of three structures, each defining an execution cache region. These regions must be located in the non-volatile program storage, for example, flash ROM) address space. Only the regions specified are enabled for cache. This means that if the host environment does not specify any executable cache regions, then no area of the flash ROM is cached, resulting in slower boot times.

FchInterface

This is a structure containing parameters related to the FCH controller hub that need to be applied at the reset time point.

```
typedef struct {
     IN OUT       UINT32      ExeCacheStartAddr;
     IN OUT       UINT32      ExeCacheSize;
} EXECUTION_CACHE_REGION;
```

ExeCacheStartAddr

Requested start address for the execution region. This start address may need to be adjusted to conform to alignment requirements. The adjusted start address is placed in this same location for inspection upon exit.

ExeCacheSize

Requested execution region size. This size may need to be adjusted to conform to alignment requirements. The adjusted size is placed in this same location for inspection upon exit. To not allocate a region, set the input parameters to zero (ExeCacheSize=0).

```
typedef struct {
  BOOLEAN     UmiGen2;
  BOOLEAN     SataEnable;
  BOOLEAN     IdeEnable;
  BOOLEAN     GppEnable;
  GPP_LINKMODE GppLinkConfig;
  BOOLEAN     Xhci0Enable;
  BOOLEAN     Xhci1Enable;
  } FCH_RESET_INTERFACE;
```

UmiGen2

This item controls whether a GEN2 data rate of UMI (Unified Media Interface, the link between chipset Northbridge and Southbridge) is enabled or disabled.

SataEnable

This item controls the SATA control function. When set to TRUE, the SATA controller function will be enabled.

IdeEnable

This item controls whether the IDE controller is to be made hidden. A FALSE value means the IDE controller is hidden and the Combined Mode is disabled, and the SATA controller has full control of all 6 ports when operating in non-IDE mode; a TRUE value means the IDE controller is exposed and the Combined Mode is enabled, the SATA controller will have control over port 0 through port 3, the IDE controller controls port 4 and 5.

GppEnable

This item controls whether the GPP ports are initialized and enumerated. A FALSE value means the ports are disabled. A TRUE value means the ports will be initialized and their devices enumerated.

GppLinkConfig

This item controls GPP Port configuration. The item is initialized by “BLDCFG_FCH_GPP_LINK_CONFIG” on page 217 and is presented here for possible modification.

Xhci0Enable

This item controls whether XHCI controller zero is enabled. A FALSE value means the controller and its ports are disabled. A TRUE value means the controller is enabled.

Xhci1Enable

This item controls whether XHCI controller one is enabled. A FALSE value means the controller and its ports are disabled. A TRUE value means the controller is enabled. In order for XHCI controller one to be enabled, controller zero must be enabled.

Description

A minimal initialization of the processor core is performed. This procedure must be called by all processor cores. The code path separates the BSP from the APs and performs a separate and appropriate list of tasks for each class of core.

For the BSP, the following actions are performed:

- Internal heap sub-system initialization
- Primary non-coherent HyperTransport™ link initialization
- Return to the host environment to test for Recovery Mode.

The AP processor cores do not participate in the recovery process; however, they execute this routine after being released to execute by the BSP during the main boot process. Their actions include the following:

- Internal heap sub-system initialization
- Proceed to a wait loop waiting for commands from the BSP

If indicated by “BLDCFG_PCI_MMIO_BASE” on page 182, the extended MMIO configuration is initialized and PCI configuration accesses will use MMIO rather than port IO.

For the cache regions, up to three regions of execution cache can be allocated following the following rules:

1. Once a region is allocated, it cannot be de-allocated. However, it can be expanded.
2. At most, two of the three regions can be located above 1 MByte. A region failing this rule is ignored.
3. All region addresses must be at or above the 0x000D0000 linear address. A region failing this rule is ignored.
4. The address is aligned on a 32-KByte boundary. Starting addresses is rounded down to the nearest 32-KByte boundary.
5. The execution cache size must be a multiple of 32 KByte. Size is rounded up to the next multiple of 32 KByte.

6. A region must not span either the 1-MByte boundary or the 4-GByte boundary. Allocated size is truncated to not span the boundary.
7. The granted cached execution regions, address, and size are calculated based on the available cache resources of the processor core. Allocations are made up to the limit of cache available on the installed processor.

Warning: Enabling instruction cache outside of this interface can cause data corruption.

Dependencies

This procedure is expected to be executed soon after a system reset for the main boot path or resume path of execution. See “AGESA™ Core Software” on page 37 for more discussion about the boot sequence.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Because the heap system is not yet operational at the point of the interface call, the host environment must allocate the storage for the `AMD_RESET_PARAMS` structure before making the first call to `AmdCreateStruct`. See the “ByHost” allocation method.

The host environment MUST NOT modify the cache controls (MTRRs, etc) during the Cache-As-Ram time period (reset through final return from AmdInitPost).

Status Codes Returned

AGESA_SUCCESS	Early initialization completed successfully.
AGESA_WARNING	One or more of the execution cache allocation rules were violated, but an adjustment was made and space was allocated.
	A BIST error was found on one of the cores.
AGESA_ERROR	One or more of the execution cache allocation rules were violated, which resulted in a requested cache region to not be allocated.
	The storage space allocated for the `AMD_RESET_PARAMS` structure is insufficient.

For WARNING or ERROR, a message is logged with data indicating which region caused the issue and which rule was violated. Please refer to “Sub-Class: Processor” on page 272.

14.2 Boot Branch Functions

AmdInitEarly

This procedure establishes the program environment needed for the host environment to start the boot sequence.

Prototype

```
AGESA_STATUS
AmdInitEarly (
     IN OUT  AMD_EARLY_PARAMS *EarlyParams
);
```

Parameters

EarlyParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      EXECUTION_CACHE_REGION  CacheRegion[3];
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      GNB_CONFIGURATION       GnbConfig;
} AMD_EARLY_PARAMS;
```

CacheRegion

This is the same as described in “AmdInitReset” on page 99. At this time point the host environment may add additional cache regions to cover the main boot code.

PlatformConfig

A structure containing platform-specific operational characteristics. This structure is initially populated by the initializer with a copy of the same structure that was created at build time using the build configuration controls.

GnbConfig

A structure containing the platform description of any special handling needs for the graphics and PCIe® subsystems and operational limitations imposed by the platform.

```
typedef struct {
     IN      PERFORMANCE_PROFILE PlatformProfile;
     IN      UINT8               CoreLevelingMode;
```

```
    IN      PLATFORM_CSTATE_MODES CStateMode;
    IN      UINT32              CStatePlatformData;
    IN      UINT16              CStateIoBaseAddress;
    IN      PLATFORM_CPB_MODES CpbMode;
    IN      BOOLEAN             UserOptionDmi;
    IN      BOOLEAN             UserOptionPState;
    IN      BOOLEAN             UserOptionSrat;
    IN      BOOLEAN             UserOptionSlit;
    IN      BOOLEAN             UserOptionWhea;
    IN      PLATFORM_LOW_POWER_PSTATE_MODES
                                LowPowerPstateForProcHot;
    IN      UINT32              PowerCeiling;
    IN      BOOLEAN             ForcePstateIndependent;
    IN      UINT32              PstatesPsdPolicy;
    IN      UINT32              CpuFrequencyLimit;;
    IN      PLATFORM_CONNECTED_STANDBY_MODES
                                CfgPlatformConnectedStandbyMode;
    IN      UINT32              NumberOfIoApics;
    IN      PLATFORM_VRM_CONFIGURATION VrmProperties[MaxVrmType];
    IN      BOOLEAN             ProcessorScopeInSb;
    IN      CHAR8               ProcessorScopeName0;
    IN      CHAR8               ProcessorScopeName1;
    IN      UINT8               GnbHdAudio;
    IN      UINT8               AbmSupport;
    IN      UINT8               DynamicRefreshRate;
    IN      UINT8               LcdBackLightControl;
    IN      UINT16              HtcTemperatureLimit;
    IN      UINT16              LhtcTemperatureLimit;
    IN      ACP_SIZE            AcpSize;
} PLATFORM_CONFIGURATION;
```

PlatformProfile

Several configuration settings for the processor depend upon other parts and general designer choices for the system. The determination of these data points is not standard for all platforms, so the host environment needs to provide these to specify how the system is to be configured.

CoreLevelingMode

Indicates how to balance the number of cores per processor. This value is initially declared by the build configuration element “BLDCFG_CORE_LEVELING_MODE” on page 188. The value is presented here for possible host environment modification.

CStateMode

The element specifies the operational mode of the C State feature. The mode in use will depend on the capabilities of the processor that is installed and the chipset on the motherboard. This value is initially declared by the build configuration element “BLDCFG_PLATFORM_CSTATE_MODE” on

page 187. The value is presented here for possible host environment modification.

CStatePlatformData

This data element will contain required operational information for the type of C State chosen by the CStateMode parameter. This value is initially declared by the build configuration element "BLDCFG_PLATFORM_CSTATE_OPDATA" on page 187. The value is presented here for possible host environment modification.

CStateIoBaseAddress

This item specifies a free block of 8 consecutive bytes of I/O ports that can be used to allow the CPU to enter C States. This value is initially declared by the build configuration element "BLDCFG_PLATFORM_CSTATE_IO_BASE_ADDRESS" on page 187. The value is presented here for possible host environment modification.

CpbMode

This item provides for forcing Core Performance Boost (CPB) to disabled. This value is initially declared by the build configuration element "BLDCFG_PLATFORM_CPB_MODE" on page 187. The value is presented here for possible host environment modification.

UserOptionDmi

When set to TRUE, the DMI data table is generated. The default is TRUE if the DMI feature is installed, else it is FALSE.

See also "BLDOPT_REMOVE_DMI" on page 179.

UserOptionPState

When set to TRUE, the PState data tables are generated. The default is TRUE if the PState feature is installed, else it is FALSE. See also "BLDOPT_REMOVE_ACPI_PSTATES" on page 178.

UserOptionSrat (deprecated)

This parameter is deprecated and will be removed in subsequent revisions.

UserOptionSlit (deprecated)

This parameter is deprecated and will be removed in subsequent revisions.

UserOptionWhea

When set to TRUE, the WHEA data table is generated. The default is TRUE if the WHEA feature is installed, else it is FALSE.See also "BLDOPT_REMOVE_WHEA" on page 179.

LowPowerPstateForProcHot

Specifies whether to create a low power P-State for power savings using processor hot signal throttling. This option should only be used on certain platforms, those which provide the signaling support in order to constrain power. See also "BLDOPT_REMOVE_LOW_POWER_STATE_FOR_PROCHOT" on page 179.

LOW_POWER_PSTATE_FOR_PROCHOT_AUTO - The low power P-State will be created if the BKDG provides for this feature (default).
LOW_POWER_PSTATE_FOR_PROCHOT_DISABLE - The low power P-State will not be created.

PowerCeiling

Specifies a maximum power usage limit for the platform. Default is zero, indicating no limit. This parameter is provided to allow end users to artificially establish a ceiling limit on the amount of power the processor core will use. This is a numerical value representing a power ceiling in milliwatts. This value is initially declared by the build configuration element "BLDCFG_AMD_TDP_LIMIT" on page 193. The value is presented here for possible host environment modification.

ForcePstateIndependent

Force P-States to be independent for each core. This value is initially declared by the build configuration element "BLDCFG_FORCE_INDEPENDENT_PSD_OBJECT" on page 186. The value is presented here for possible host environment modification.

CpuFrequencyLimit

Specifies the maximum frequency that the CPU cores are allowed to use. This value is initially declared by the build configuration element "BLDCFG_CPU_FREQUENCY_LIMIT" on page 186. The value is presented here for possible host environment modification.

CfgPlatformConnectedStandbyMode

Specifies wheather or not the Connected Standby feature is to be enabled. This value is initially declared by the build configuration element "BLDCFG_CPU_CONECTED_STANDBY_MODE" on page 187. The value is presented here for possible host environment modification.

NumberOfIoApics

Specifies the number of IO APICs in the system. This value is initially declared by the build configuration element "BLDCFG_PLATFORM_NUM_IO_APICS" on page 186. The value is presented here for possible host environment modification.

VrmProperties

Collects platform voltage regulator module (VRM) properties that are required for power management.

ProcessorScopeInSb

This value is initially declared by the build configuration element "BLDCFG_PROCESSOR_SCOPE_IN_SB" on page 186. The value is presented here for possible host environment modification.

ProcessorScopeName0

This value is initially declared by the build configuration element "BLDCFG_PROCESSOR_SCOPE_NAME0" on page 186. The value is presented here for possible host environment modification.

ProcessorScopeName1

This value is initially declared by the build configuration element "BLDCFG_PROCESSOR_SCOPE_NAME1" on page 186. The value is presented here for possible host environment modification.

GnbHdAudio

This item customizes the GFX High Definition (HD) audio controller. This value is initially declared by the build configuration element "BLDCFG_CFG_GNB_HD_AUDIO" on page 197. The value is presented here for possible host environment modification.

AbmSupport

This item customizes the Ambient Brightness Monitor (ABM) back light control for LVDS and eDP. This value is initially declared by the build configuration element "BLDCFG_CFG_ABM_SUPPORT" on page 197. The value is presented here for possible host environment modification.

DynamicRefreshRate

This item specifies the minimum refresh rate in Hz for the embedded display panel. This value is initially declared by the build configuration element "BLDCFG_CFG_DYNAMIC_REFRESH_RATE" on page 198. The value is presented here for possible host environment modification.

LcdBackLightControl

This item customizes support for Pulse Width Modulation (PWM) backlight control. This value is initially declared by the build configuration element "BLDCFG_CFG_LCD_BACK_LIGHT_CONTROL" on page 198. The value is presented here for possible host environment modification.

HtcTemperatureLimit

This item customizes support for thermal control. This value is initially declared by the build configuration element "BLDCFG_HTC_TEMPERATURE_LIMIT" on page 203. The value is presented here for possible host environment modification.

LhtcTemperatureLimit

This item customizes support for thermal control. This value is initially declared by the build configuration element "BLDCFG_LHTC_TEMPERATURE_LIMIT" on page 203. The value is presented here for possible host environment modification.

AcpSize

This item customizes support for Audio Co-Processor (ACP). This value is initially declared by the build configuration element "BLDCFG_ACP_SIZE" on page 198. The value is presented here for possible host environment modification.

```
typedef struct {
    IN      PLATFORM_CONTROL_FLOW   PlatformControlFlowMode;
    IN      BOOLEAN                 Use32ByteRefresh;
    IN      BOOLEAN                 UseVariableMctIsocPriority;
    IN      ADVANCED_PERFORMANCE_PROFILE AdvancedPerformanceProfile;
    IN      PLATFORM_POWER_POLICY   PlatformPowerPolicy;
    IN      BOOLEAN                 NbPstatesSupported;
} PERFORMANCE_PROFILE;
```

PlatformControlFlowMode

This value is used to select the optimum flow control method for the platform. This value is initially declared by the build configuration element "BLDCFG_PLATFORM_CONTROL_FLOW_MODE" on page 191. The value is presented here for possible host environment modification.

UseVariableMctIsocPriority

System chipsets may employ the Isochronous data channels for special packets to improve performance. This value is initially declared by the build configuration element "BLDCFG_USE_VARIABLE_MCT_ISOC_PRIORITY" on page 192. The value is presented here for possible host environment modification.

Use32ByteRefresh

This feature provides the refresh packet size to better optimize the Isochronous data channel performance. This value is initially declared by the build

configuration element “BLDCFG_USE_32_BYTE_REFRESH” on page 191. The value is presented here for possible host environment modification.

AdvancedPerformanceProfile

These settings provide for performance tuning to optimize for specific workloads. For general performance use the recommended settings.

```
typedef struct {
      IN    HARDWARE_PREFETCH_MODE  HardwarePrefetchMode;
      IN    SOFTWARE_PREFETCH_MODE  SoftwarePrefetchMode;
      IN    DRAM_PREFETCH_MODE      DramPrefetchMode;
} ADVANCED_PERFORMANCE_PROFILE;
```

HardwarePrefetchMode

This feature provides advanced tuning of the hardware prefetcher. This value is initially declared by the build configuration element “BLDCFG_PERFORMANCE_HARDWARE_PREFETCHER” on page 192. The value is presented here for possible host environment modification.

SoftwarePrefetchMode

This feature provides advanced tuning of software prefetches. This value is initially declared by the build configuration element “BLDCFG_PERFORMANCE_SOFTWARE_PREFETCHES” on page 192. The value is presented here for possible host environment modification.

DramPrefetchMode

This feature provides advanced tuning of the DRAM prefetcher. This value is initially declared by the build configuration element “BLDCFG_PERFORMANCE_DRAM_PREFETCHER” on page 193. The value is presented here for possible host environment modification.

NbPstatesSupported

This feature provides an indication of desired processor NB P-States setting. This value is initially declared by the build configuration element “BLDCFG_NB_PSTATES_SUPPORTED” on page 193. The value is presented here for possible host environment modification.

```
typedef struct {
  IN UINT32  CurrentLimit;
  IN UINT32  LowPowerThreshold;
  IN UINT32  SlewRate;
  IN BOOLEAN HiSpeedEnable;
  IN UINT32  MaximumCurrentLimit;
  IN UINT16  SviOcpLevel;
```

```
} PLATFORM_VRM_CONFIGURATION;

typedef enum  {
  CoreVrm,
  NbVrm,
  MaxVrmType
} PLATFORM_VRM_TYPE;
```

The platform configuration contains an array of PLATFORM_VRM_CONFIGURATION, with one structure for each voltage plane. The first element, CoreVrm, contains the VRM properties for the CPU Cores. The second element, NbVrm, contains the VRM properties for the processor Northbridge. Each member of PLATFORM_VRM_CONFIGURATION has a build configuration item for each of the two VRMs. Use the build configuration item corresponding to either the CoreVrm or NbVrm to set the VRM properties for each. Not all processors support all possible settings; see processor specific documentation for details.

CurrentLimit

Provides the VRM current limit. This value is initially declared by the build configuration element "BLDCFG_VRM_CURRENT_LIMIT" on page 183 for the CoreVrm property set, or by "BLDCFG_VRM_NB_CURRENT_LIMIT" on page 183 for the NbVrm property set. The value is presented here for possible host environment modification.

LowPowerThreshold

Provides information about the VRM's low power mode. This value is initially declared by the build configuration element "BLDCFG_VRM_LOW_POWER_THRESHOLD" on page 183 for the CoreVrm property set, or by "BLDCFG_VRM_NB_LOW_POWER_THRESHOLD" on page 184 for the NbVrm property set. The value is presented here for possible host environment modification.

SlewRate

Provides the rate at which the VRM can transition. This value is initially declared by the build configuration element "BLDCFG_VRM_SLEW_RATE" on page 184 for the CoreVrm property set, or by "BLDCFG_VRM_NB_SLEW_RATE" on page 184 for the NbVrm property set. The value is presented here for possible host environment modification.

HiSpeedEnable

The VRM bus interface is high speed. This value is initially declared by the build configuration element "BLDCFG_VRM_HIGH_SPEED_ENABLE" on page 184 for the CoreVrm property set, or by "BLDCFG_VRM_NB_HIGH_SPEED_ENABLE" on page 185 for the

NbVrm property set. The value is presented here for possible host environment modification.

MaximumCurrentLimit

Provides maximum current limit for the VRM. This value is initially declared by the build configuration element "BLDCFG_VRM_MAXIMUM_CURRENT_LIMIT" on page 185 for the CoreVrm property set, or by "BLDCFG_VRM_NB_MAXIMUM_CURRENT_LIMIT" on page 185 for the NbVrm property set. The value is presented here for possible host environment modification.

SviOcpLevel

Provides the current level at which over current protection (OCP) is initiated by the VRM. This value is initially declared by the build configuration element "BLDCFG_VRM_SVI_OCP_LEVEL" on page 185 for the CoreVrm property set, or by "BLDCFG_VRM_NB_SVI_OCP_LEVEL" on page 185 for the NbVrm property set. The value is presented here for possible host environment modification.

```
typedef struct {
  IN       PCIe_COMPLEX_DESCRIPTOR  *PcieComplexList;
  IN       UINT8                    PsppPolicy;
} GNB_CONFIGURATION;
```

For processors with PCIe® support, these items provide customization of PCIe® related settings. See "Graphics Northbridge Details" on page 325 for more details.

PcieComplexList

Provide all customizations for the PCIe® topology.

PsppPolicy

Provide the desired performance vs. power policy.

```
typedef struct {
  IN       UINT32               Flags;
  IN       UINT32               SocketId;
  IN       PCIe_PORT_DESCRIPTOR *PciePortList;
  IN       PCIe_DDI_DESCRIPTOR  *DdiLinkList;
  IN       VOID                 *Reserved;
} PCIe_COMPLEX_DESCRIPTOR;
```

Flags

Indicates if this is the last descriptor in the list.

SocketId

The socket ID for the processor which hosts this PCIe® topology.

PciePortList

A list of PCIe ports and their customization data.

DdiLinkList

A list of DDI links and their customization data.

Description

A full initialization of the processor is performed. Action details differ for the BSP and AP processor cores.

For the BSP, the following actions are performed:

- Full HyperTransport™ link initialization, coherent and non-coherent
- Processor register loading
- Microcode patch load
- Errata workaround processing
- Launch all processor cores
- Configure the processor power management capabilities
- Request a warm reset if needed

For the AP, the following actions are performed:

- processor register loading
- microcode patch load
- errata workaround processing
- configure the processor power management capabilities

Dependencies

This procedure is expected to be called before main memory initialization and before the system warm reset. Prior to this, the basic configuration done by the AmdInitReset routine must be completed.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

The processes performed at this time point require communication between processor cores. The host environment must recognize that all processor cores are running in parallel and avoid activities that might interfere with the core-to-core communication, such as modifying the MTRR settings or writing to the APIC registers.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_WARNING	One of more of the allocation rules were violated, but an adjustment was made and space was allocated.
AGESA_ERROR	One or more of the allocation rules were violated, which resulted in a requested cache region to not be allocated.
AGESA_FATAL	A processor is detected which has no family support installed in the current build.

AmdInitPost

Initialize system memory. Transition boot from pre-memory phase to post-memory phase.

Prototype

```
AGESA_STATUS
AmdInitPost (
     IN OUT  AMD_POST_PARAMS         *PostParams
);
```

Parameters

PostParams

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      MEM_PARAMETER_STRUCT    MemConfig;
     IN      GNB_POST_CONFIGURATION  GnbPostConfig;
} AMD_POST_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics.This is the same structure described in “AmdInitEarly” on page 103.

MemConfig

A structure containing platform configuration descriptions pertaining to the memory channels.

```
typedef struct  {
     // Basic (Return parameters)
     // This section contains the outbound status of the memory init
      OUT    BOOLEAN           GStatus[GsbEOL];
      OUT    UINT32            HoleBase;
      OUT    UINT32            Sub4GCacheTop;
      OUT    UINT32            Sub1THoleBase;
      OUT    UINT32            SysLimit;
      OUT    DIMM_VOLTAGE      DDR3Voltage;
      OUT    VDDP_VDDR_VOLTAGE VddpVddrVoltage;
      OUT    UINT8             ExternalVrefValue;
      OUT    MEM_DATA_STRUCT   *MemData;
     // Memory Map/Mgt.
     IN      UINT16            BottomIo;
     IN      BOOLEAN           LimitMemoryToBelow1Tb;
     // Dram Timing
```

```
    IN      USER_MEMORY_TIMING_MODE UserTimingMode;
    IN      MEMORY_BUS_SPEED  MemClockValue;
    // Dram Configuration
    IN      BOOLEAN           EnableBankIntlv;
    IN      BOOLEAN           EnableNodeIntlv;
    IN      BOOLEAN           EnableChannelIntlv;
    IN      BOOLEAN           EnableDllPDBypassMode;
    IN      BOOLEAN           EnableEccFeature;
    IN      BOOLEAN           EnablePowerDown;
    IN      BOOLEAN           EnableOnLineSpareCtl;
    IN      UINT8             *TableBasedAlterations;
    IN      PSO_TABLE         *PlatformMemoryConfiguration;
    IN      BOOLEAN           EnableParity;
    IN      BOOLEAN           EnableBankSwizzle;
    IN      BOOLEAN           EnableMemClr;
    IN      UMA_MODE          UmaMode;
    IN OUT  UINT32            UmaSize;
       OUT  UINT64            UmaBase;
    IN      BOOLEAN           MemRestoreCtl;
    IN      BOOLEAN           SaveMemContextCtl;
    IN OUT  AMD_S3_PARAMS     MemContext;
    IN      BOOLEAN           IsCapsuleUpdate;
    IN      BOOLEAN           ExternalVrefCtl;
    IN      FORCE_TRAIN_MODE  ForceTrainMode;
    IN      TECHNOLOGY_TYPE   DimmTypeUsedInMixedConfig;
    IN      BOOLEAN           AmpEnable;
    IN      BOOLEAN           AmpWarningMsgEnable;
       OUT  AMP_STATUS        AmpStatus;
       OUT  AMP_DIMM_VOLTAGE  AmpVoltage;
    IN      BOOLEAN           DramDoubleRefreshRate;
    IN      PMU_TRAIN_MODE    PmuTrainMode;
} MEM_PARAMETER_STRUCT;
```

Note: some minor details may differ from the implementation. Please see the internal documentation for full details.

GStatus

Global Status. An array of indicators of various conditions found during memory initialization. The full definition of indicators can be found in the AGESA.H file.

HoleBase

This value is associated with BottomIO and usually has the same value as BottomIO. However, conditions can occur where the BottomIO value must be changed and then this value reflects the actual base used. They may occur when new PCI cards are installed, which require more IO space.

The value represents address bits 8 through 39 of the system address where the sub 4-Gbyte DRAM hole for HW remapping begins (BASE[39:8]). A value of zero indicates that the remapping is disabled.

Sub4GCacheTop

This value represents the address of the end of the memory zone located below the 4-Gbyte boundary that is able to be cached. This may or may not be same as HoleBase or BottomIO.

The value is the address of the last byte of cached memory below the 4-Gbyte boundary.

Sub1THoleBase

This value represents address bits 16 through 47 of the system address where the memory hole located below the 1 terabyte (TByte) boundary begins. This output is only valid if SysLimit is greater than 1 TByte.

SysLimit

This value represents address bits 8 through 39 of the system address where the last byte of physical system memory is located. (LIMIT[39:8])

Ddr3Voltage

This output provides the DDR3 voltage for the system.

VddpVddrVoltage

This item specifies the voltage to be supplied to the memory PHY power pins. If the `IsValid` element is TRUE, the host platform is expected to adjust the VRM to supply the indicated voltage. This check should be made during processing of the AgesaCallOut AgesaHookBeforeDramInit.

```
typedef struct {
      BOOLEAN IsValid;
      MEMORY_PHY_VOLTAGE Voltage; BOOLEANAmpVoltageValid;
} VDDP_VDDR_VOLTAGE;
```

IsValid

Specifies if the memory PHY voltage in the next parameter is to be used.

TRUE - Valid, make the change. FALSE - Do not change voltage.

Voltage

Specifies the memory PHY voltage to be used. For values, see “BLDCFG_MEMORY_PHY_VOLTAGE” on page 210.

MemData

This is a pointer to an internal data structure used by the memory initialization code. This pointer is provided for use by expert diagnosticians only.

LimitMemoryToBelow1Tb

This item allows memory to be limited below 1 TByte, if needed for certain operating systems. This value is initially declared by the build configuration

element “BLDCFG_LIMIT_MEMORY_TO_BELOW_1TB” on page 213. The value is presented here for possible host environment modification.

EnableBankIntlv

Enables the system to use the DRAM bank (also known as chip-select) interleaving feature. This value is initially declared by the build configuration element “BLDCFG_MEMORY_ENABLE_BANK_INTERLEAVING” on page 206. The value is presented here for possible host environment modification.

EnableNodeIntlv

Enables the system to use the memory node interleaving feature. This value is initially declared by the build configuration element “BLDCFG_MEMORY_ENABLE_NODE_INTERLEAVING” on page 206. The value is presented here for possible host environment modification.

EnableChannelIntlv

Enables the system to use the memory channel interleaving feature. This value is initially declared by the build configuration element “BLDCFG_MEMORY_CHANNEL_INTERLEAVING” on page 207. The value is presented here for possible host environment modification.

EnableOnLineSpareCtl

The item controls the operation of the On-line Spare feature. This feature is recommended only for expert users and is described in the *BIOS and Kernel Developer's Guides (BKDG)*.

This value is initially declared by the build configuration element “BLDCFG_ONLINE_SPARE” on page 207. The value is presented here for possible host environment modification.

EnableParity

Use parity error detection if available on the DIMMs.

This value is initially declared by the build configuration element “BLDCFG_MEMORY_PARITY_ENABLE” on page 208. The value is presented here for possible host environment modification.

EnableBankSwizzle

This value is initially declared by the build configuration element “BLDCFG_BANK_SWIZZLE” on page 208. The value is presented here for possible host environment modification.

EnableMemClr

This value is initially provided based on installed processor support. The value is presented here for possible host environment modification.

UserTimingMode

Selects the timing mode.

This value is initially declared by the build configuration element "BLDCFG_TIMING_MODE_SELECT" on page 208. The value is presented here for possible host environment modification.

MemClockValue

Selects the memory-clock frequency.

This value is initially declared by the build configuration element "BLDCFG_MEMORY_CLOCK_SELECT" on page 208. The value is presented here for possible host environment modification.

EnableDllPDBypassMode

Enables low power mode for soldered down memory systems which follow the low power guidelines. This value is initially declared by the build configuration element "BLDCFG_DDR_PHY_DLL_BYPASS_MODE" on page 210. The value is presented here for possible host environment modification.

EnableEccFeature

Enables the system to use the ECC feature. This value is initially declared by the build configuration element "BLDCFG_ENABLE_ECC_FEATURE" on page 210. The value is presented here for possible host environment modification.

EnablePowerDown

This is a power conservation option whose value is initially declared by the build configuration element "BLDCFG_MEMORY_POWER_DOWN" on page 207. The value is presented here for possible host environment modification.

TableBasedAlterations

This is a pointer to an array of data bytes describing desired modifications to register settings. The format of this data table is described in "Memory Details" on page 274.

PlatformMemoryConfiguration

This is a pointer to a table that contains platform specific settings such as MemClk routing and the number of DIMM slots per channel. By default this will point to default conservative settings. This is presented here for possible

host environment modification. The format of this data table is described in "Platform Specific Override" on page 277.

UmaMode

Specifies whether to allocate memory for UMA support. The value is initially declared by the build configuration element "BLDCFG_UMA_ALLOCATION_MODE" on page 212. The value is presented here for possible host environment modification.

UmaSize

The amount of memory requested. This is updated to reflect the amount of memory allocated. The value is initially declared by the build configuration element "BLDCFG_UMA_ALLOCATION_SIZE" on page 213. The value is presented here for possible host environment modification.

UmaBase

This output provides the 64 bit base address of the allocated UMA region.

MemRestoreCtl

Specifies whether to restore previously saved training configuration rather than performing training.

SaveMemContextCtl

Specifies whether to update the memory context output with the training configuration. This data may be saved in non-volatile storage.

MemContext

The memory context to either restore or output.

IsCapsuleUpdate

Specifies that the current boot has capsule data contained in the system memory. This will prevent the capsule data from being cleared during memory initialization.

FALSE - This is not a capsule reboot (default).
TRUE - This is a capsule reboot.

ExternalVrefCtl

Specifies the control of external Vref for 2D memory training.The value is initially declared by the build configuration element "BLDCFG_ENABLE_EXTERNAL_VREF_FEATURE" on page 213. The value is presented here for possible host environment modification.

ForceTrainMode

Specifies whether to force memory training to the specified mode.

0 - Force 1D memory training for all configurations.
1 - Force 2D memory training for all configurations.
2 - Auto, the memory training mode will be selected automatically (default).

DimmTypeUsedInMixedConfig

Specifies how to handle installations of mixed memory technology. The value is initially declared by the build configuration element "BLDCFG_DIMM_TYPE_USED_IN_MIXED_CONFIG" on page 213. The value is presented here for possible host enviroment modification.

AmpEnable

Specifies whether AMP memory overclocking is to be enabled.

TRUE - Enabled. FALSE - Disabled (default).

AmpWarningMsgEnable

Specifies to provide warning events if the AMP capability does not match the desired AMP mode.

TRUE - Generate warning events to the event log.
FALSE - Suppress warning events (default).

AmpStatus

Reports whether AMP memory overclocking is enabled. Warning status is also provided for error reporting. The warning status is set here regardless of the state of AmpWarningMsgEnable.

```
typedef struct {
      BOOLEAN     AmpVoltageValid;
      BOOLEAN     AmpSupportDetectedButNotEnabled;
      BOOLEAN     AmpSelectedButNotEnabled;
} AMP_STATUS;
```

AmpVoltageValid

The memory voltage is reported in AmpVoltage.

TRUE - Valid. FALSE - Use Ddr3Voltage.

AmpSupportDetectedButNotEnabled

AMP support was detected, but not enabled.

TRUE - Warning present. FALSE - No warning.

AmpSelectedButNotEnabled

AMP support was selected, but could not be enabled.

TRUE - Warning present. FALSE - No warning.

AmpVoltage

When the value of AmpStatus.AmpVoltageValid is TRUE, this specifies the memory DIMM AMP voltage.

DramDoubleRefreshRate

Specifies if the calculated memory refresh rate should be doubled (i.e. time period between refresh cycles cut in half).

PmuTrainMode

Specifies the memory data bus training mode which will be performed by the PMU. The value is initially declared by the build configuration element "BLDCFG_PMU_TRAINING_MODE" on page 214. The value is presented here for possible host enviroment modification.

```
typedef struct {
    IN      UINT8       IgpuEnableDisablePolicy;
} GNB_POST_CONFIGURATION;
```

IgpuEnableDisablePolicy

Specifies whether to enable or disable the processor's internal graphics when discrete graphics cards are present. The value is initially declared by the build configuration element "BLDCFG_IGPU_ENABLE_DISABLE_POLICY" on page 204. The value is presented here for possible host environment modification.

Description

The main system memory is located, initialized, and brought on-line. The processors are prepared for full operation and control by the host environment. Action details differ for the BSP and AP processor cores.

For the BSP, the following actions are performed:

- Full memory initialization and configuration. BSP is the master for this process and may delegate some tasks to APs.
- Check the BIST status of the BSP
- AP collection of data for use later.
- Transfer the HOBs including the artifact data out of the pre-memory cache storage into a temporary holding buffer in the main memory.
- Shut down the APs.
- Prepare for the host environment to begin main boot activity.
- Disable the pre-memory stack.

For the APs, the following actions are performed:

– Check the BIST status of the AP
– Report core identity information.
– Execute indicated memory initialization processes as directed.
– Disable the pre-memory stack.
– Prepare to halt, giving control to host environment.

The exit state for the APs is described as follows:

- Memory region 00000—9FFFF MTRRS are set as WB memory.
- Memory region A0000—DFFFF MTRRS are set as UC IO.
- Memory region E0000—FFFFF MTRRS are set as UC memory.
- MTRRs used for execution cache are cleared.
- Processor Cache is disabled (CD bit is set).
- Top-of-Memory (TOM) set to the system top of memory as determined by the memory initialization routines.
- System lock command is enabled.
- Any family-specific clean-up done.

The entire range of system memory is enabled for Write-Back cache.

The fixed MTRRs and the variable MTRRs[7:6] are not changed in order to leave in place any flash ROM region currently set for Write-Protect execution cache.

Dependencies

This procedure is called after the host environment has determined that a normal boot to operating system should be performed after any system warm reset is completed and after the configuration done by AmdInitEarly has completed.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

The processes performed at this time point require communication between processor cores. The host environment must recognize that all processor cores are running in parallel and avoid activities that might interfere with the core-to-core communication, such as modifying the MTRR settings or writing to the APIC registers.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_ALERT	A BIST Error was observed.
AGESA_WARNING	
AGESA_ERROR	
AGESA_FATAL	

AmdInitEnv

This procedure closes down the pre-memory operations and transfers any artifact data into main memory.

Prototype

```
AGESA_STATUS
AmdInitEnv (
     IN OUT  AMD_ENV_PARAMS          *EnvParams
);
```

Parameters

EnvParams

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      GNB_ENV_CONFIGURATION   GnbEnvConfiguration;
     IN      FCH_INTERFACE           FchInterface;
} AMD_ENV_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics.This is the same structure described in “AmdInitEarly” on page 103.

GnbEnvConfiguration

A structure containing all the required input configuration settings for GNB initialization at the AmdInitEnv stage.

FchInterface

A structure containing all the required input configuration settings for FCH initialization at the AmdInitEnv stage.

```
typedef struct {
     IN UINT8      Gnb3dStereoPinIndex;
     IN BOOLEAN    IommuSupport;
     IN UINT16     LvdsSpreadSpectrum;
     IN UINT16     LvdsSpreadSpectrumRate;
     IN UINT8      LvdsPowerOnSeqDigonToDe;
     IN UINT8      LvdsPowerOnSeqDeToVaryBl;
     IN UINT8      LvdsPowerOnSeqDeToDigon;
     IN UINT8      LvdsPowerOnSeqVaryBlToDe;
     IN UINT8      LvdsPowerOnSeqOnToOffDelay;
     IN UINT8      LvdsPowerOnSeqVaryBlToBlon;
     IN UINT8      LvdsPowerOnSeqBlonToVaryBl;
```

```
    IN UINT16     LvdsMaxPixelClockFreq;
    IN UINT32     LcdBitDepthControlValue;
    IN UINT8      Lvds24bbpPanelMode;
    IN LVDS_MISC_CONTROL LvdsMiscControl;
    IN UINT16     PcieRefClkSpreadSpectrum;
    IN BOOLEAN    GnbRemoteDisplaySupport;
    IN UINT8      LvdsMiscVoltAdjustment;
    IN DISPLAY_MISC_CONTROL DisplayMiscControl;
    IN UINT32     GpuFrequencyLimit;
} GNB_ENV_CONFIGURATION;
```

Gnb3dStereoPinIndex

This item configures which pin the platform uses for Stereo 3D. This value is initially declared by the build configuration element "BLDCFG_STEREO_3D_PINOUT" on page 198. The value is presented here for possible host environment modification.

IommuSupport

This item configures IOMMU support. The value is initially declared by the build configuration element "BLDCFG_IOMMU_SUPPORT" on page 199. The value is presented here for possible host modification.

LvdsSpreadSpectrum

This item configures LVDS spread spectrum support. The value is initially declared by the build configuration element "BLDCFG_GFX_LVDS_SPREAD_SPECTRUM" on page 200. The value is presented here for possible host modification.

LvdsSpreadSpectrumRate

This item configures the LVDS spread spectrum frequency. The value is initially declared by the build configuration element "BLDCFG_GFX_LVDS_SPREAD_SPECTRUM_RATE" on page 200. The value is presented here for possible host modification.

LvdsPowerOnSeqDigonToDe

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWER_ON_SEQ_DIGON_TO_DE" on page 200. The value is presented here for possible host modification.

LvdsPowerOnSeqDeToVaryBl

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWER_ON_SEQ_DE_TO_VARY_BL" on page 200. The value is presented here for possible host modification.

LvdsPowerOnSeqDeToDigon

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWER_ON_SEQ_DE_TO_DIGON" on page 200. The value is presented here for possible host modification.

LvdsPowerOnSeqVaryBlToDe

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWERS_ON_SEQ_VARY_BL_TO_DE" on page 201. The value is presented here for possible host modification.

LvdsPowerOnSeqOnToOffDelay

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWER_ON_SEQ_ON_TO_OFF_DELAY" on page 201. The value is presented here for possible host modification.

LvdsPowerOnSeqVaryBlToBlon

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWER_ON_SEQ_VARY_BL_TO_BLON" on page 201. The value is presented here for possible host modification.

LvdsPowerOnSeqBlonToVaryBl

This item configures panel initialization timing. The value is initially declared by the build configuration element "BLDCFG_LVDS_POWER_ON_SEQ_BLON_TO_VARY_BL" on page 201. The value is presented here for possible host modification.

LvdsMaxPixelClockFreq

This item configures the maximum pixel clock frequency supported. The value is initially declared by the build configuration element "BLDCFG_LVDS_MAX_PIXEL_CLOCK_FREQ" on page 201. The value is presented here for possible host modification.

LcdBitDepthControlValue

This item configures the LCD bit depth control settings. The value is initially declared by the build configuration element "BLDCFG_LCD_BIT_DEPTH_CONTROL_VALUE" on page 201. The value is presented here for possible host modification.

Lvds24bppPanelMode

This item configures the LVDS 24 BBP mode. The value is initially declared by the build configuration element

"BLDCFG_LVDS_24BBP_PANEL_MODE" on page 202. The value is presented here for possible host modification.

LvdsMiscControl

This item configures LVDS miscellaneous support settings. This value is initially declared by the build configuration elements "BLDCFG_LVDS_MISC_888_FPDI_MODE" on page 202, "BLDCFG_LVDS_MISC_DL_CH_SWAP" on page 202, "BLDCFG_LVDS_MISC_VSYNC_ACTIVE_LOW" on page 202, "BLDCFG_LVDS_MISC_HSYNC_ACTIVE_LOW" on page 202, "BLDCFG_LVDS_MISC_BLON_ACTIVE_LOW" on page 202, and "BLDCFG_LVDS_MISC_VOLT_OVERWRITE_ENABLE" on page 202. The value is presented here for possible host modification.

PcieRefClkSpreadSpectrum

This provides the spread spectrum setting for the PCIe® reference clock, when the platform has enabled spread spectrum. This value is initially declared by the build configuration element "BLDCFG_PCIE_REFCLK_SPREAD_SPECTRUM" on page 204. The value is presented here for possible host modification.

GnbRemoteDisplaySupport

This item configures remote wireless display support. The value is initially declared by the build configuration element "BLDCFG_REMOTE_DISPLAY_SUPPORT" on page 204. The value is presented here for possible host modification.

LvdsMiscVoltAdjustment

This item configures output voltage adjustment. An adjustment may be needed for LVDS via Travis converters. This value is initially declared by the build configuration element "BLDCFG_LVDS_MISC_VOLT_ADJUSTMENT" on page 202. The value is presented here for possible host modification.

DisplayMiscControl

This item configures miscellaneous display settings. This value is initially declared by the build configuration element "BLDCFG_DISPLAY_MISC_VBIOS_FAST_BOOT_ENABLE" on page 204. The value is presented here for possible host modification.

GpuFrequencyLimit

This item defines the maximum GPU frequency in MHz. This value is used to disable GPU power states that would exceed this limit. This value is initially declared by the build configuration element "BLDCFG_DISPLAY_MISC_VBIOS_FAST_BOOT_ENABLE" on page 204. The value is presented here for possible host modification.

```
typedef struct _FCH_INTERFACE {
  SD_MODE      SdConfig;
  HDA_CONFIG   AzaliaController;
  IR_CONFIG    IrConfig;
  BOOLEAN      UmiGen2;
  SATA_CLASS   SataClass;
  BOOLEAN      SataEnable;
  BOOLEAN      IdeEnable;
  BOOLEAN      SataIdeMode;
  BOOLEAN      Ohci1Enable;
  BOOLEAN      Ohci2Enable;
  BOOLEAN      Ohci3Enable;
  BOOLEAN      Ohci4Enable;
  BOOLEAN      GppEnable;
  GPP_LINKMODE GppLinkConfig;
  BOOLEAN      FchPowerFail;
} FCH_INTERFACE;
```

SdConfig

Defines the Secure Digital (SD) controller mode.

AzaliaController

Azalia HD audio controller function, ENABLED/DISABLED/AUTO.

IrConfig

Infrared (IR) operation mode.

UmiGen2

This item controls whether a GEN2 data rate of UMI (Unified Media Interface, the link between chipset Northbridge and Southbridge) is enabled or disabled.

SataClass

This item defines the target the SATA controller operating mode.

SataEnable

This item controls the SATA control function. When set to TRUE, the SATA controller function will be enabled.

IdeEnable

This item controls whether the IDE controller is to be made hidden. A FALSE value means the IDE controller is hidden and the Combined Mode is disabled, and the SATA controller has full control of all 6 ports when operating in non-IDE mode; a TRUE value means the IDE controller is exposed and the Combined Mode is enabled, the SATA controller will have control over port 0 through port 3, the IDE controller controls port 4 and 5.

SataIdeMode

This item selects between native IDE mode and legacy IDE mode.

Ohci1Enable

This item enables or disables OHCI1 controller.

Ohci2Enable

This item enables or disables OHCI2 controller.

Ohci3Enable

This item enables or disables OHCI3 controller.

Ohci4Enable

This item enables or disables OHCI4 controller.

GppEnable

This item controls whether the GPP ports are initialized and enumerated. A FALSE value means the ports are disabled. A TRUE value means the ports will be initialized and their devices enumerated.

GppLinkConfig

This item controls GPP Port configuration. The item is initialized by "BLDCFG_FCH_GPP_LINK_CONFIG" on page 217 and is presented here for possible modification.

FchPowerFail

This item controls the action by the FCH when power is resumed after a power failure. A value of zero means to power the platform off when power resumes. A value of one means to power the platform on when power resumes. A value of three means to resume to the previous state.

Description

This procedure uses the AgesaAllocateBuffer call-out to acquire permanent buffer space for the UEFI Hand-Off Blocks (HOBs). This is also known as, or includes, artifact data being used by the AGESA™ software. Upon entry to this procedure, the data is being held in a temporary memory location and it must be moved to a location controlled and protected by the host environment.

These actions are performed by the BSP. The APs are not assigned any tasks at this time point.

Dependencies

This procedure must be called after full memory is initialized and the host environment has taken control of main memory allocation. This procedure should be called before the PCI

enumeration takes place and as soon as possible after the host environment memory allocation sub-system has started.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_ERROR	The artifact data could not be found or the host environment failed to allocate sufficient buffer space.

AmdInitMid

Prepare the graphics sub-system.

Prototype

```
AGESA_STATUS
AmdInitMid (
     IN OUT  AMD_MID_PARAMS          *MidParams
);
```

Parameters

MidParams

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      GNB_MID_CONFIGURATION   GnbMidConfiguration;
     IN      FCH_INTERFACE           FchInterface;
} AMD_MID_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics.This is the same structure described in “AmdInitEarly” on page 103.

GnbMidConfiguration

A structure containing GNB parameters.

FchInterface

A structure containing FCH parameters. This is the same structure described in “AmdInitEnv” on page 124.

```
typedef struct {
     IN      UINT8       iGpuVgaMode;
     IN      UINT8       PcieAcsCapability;
     IN      UINT64      GnbIoapicAddress;
     IN      UINT8       MaxNumAudioEndpoints;
} GNB_MID_CONFIGURATION;
```

iGpuVgaMode

This specifies whether the processor internal GPU will handle resources for the primary VGA.

PcieAscCapability

This specifies whether Access Control Services (ACS) capabilities are exposed to the operating system. ACS can be used by hypervisors and may also be needed for certain compliance testing.

PcieAcsDisabled - Do not expose root bridge ACS (default).
PcieAcsEnabled - Expose root bridge ACS.

GnbIoapicAddress

The base address for the IOAPIC in the GNB may be programmed by the platform BIOS or by AGESA. If this value (GnbIoapicAddress) is NULL, then AGESA expects platform BIOS to program the base address of the IOAPIC. If this value is not NULL, then AGESA will use this value to set the base address of the IOAPIC.

MaxNumAudioIndpoints

A number specifying the number of active audio endpoints desired for this platform. The value used will be the minimum of: <this parameter>, BLDCFG_MAX_NUM_AUDIO_ENDPOINTS and the number of endpoints supported by the APU.

Description

This procedure call performs special configuration requirements for the graphics display hardware.

These actions are performed by the BSP. The APs are not assigned any tasks at this time point.

Dependencies

This procedure must be called after PCI enumeration has allocated resources, but before the video BIOS call is performed.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_ALERT	
AGESA_WARNING	

AmdInitLate

Perform initialization and configuration duties late in the boot sequence, filling information tables for use by the operating system.

Prototype

```
AGESA_STATUS
AmdInitLate (
     IN OUT  AMD_LATE_PARAMS *LateParams
);
```

Parameters

LateParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      IOMMU_EXCLUSION_RANGE_DESCRIPTOR
                                     *IvrsExclusionRangeList;
      OUT    DMI_INFO                *DmiTable;
      OUT    VOID                    *AcpiPState;
      OUT    VOID                    *AcpiSrat;
      OUT    VOID                    *AcpiSlit;
      OUT    VOID                    *AcpiWheaMce;
      OUT    VOID                    *AcpiWheaCmc;
      OUT    VOID                    *AcpiAlib;
      OUT    VOID                    *AcpiIvrs;
      OUT    VOID                    *AcpiCrat;
      OUT    VOID                    *AcpiCdit;
     IN      GNB_LATE_CONFIGURATION  GnbLateConfiguration;
} AMD_LATE_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics.This is the same structure described in “AmdInitEarly” on page 103.

IvrsExclusionRangeList

Describe the exclusion ranges which need to be reported in the ACPI IVRS table. See “IOMMU Exclusion Range Descriptor” on page 331 for details.

DmiTable

Pointer to the buffer area that contains the DMI information. The buffer is allocated by using the Call-Out AgesaAllocateBuffer.

Note: *The DMI information does not include handle fields, since it is expected that the host environment must assign handle values. There is one exception. In order to correctly associate Type 20 structures, Memory Device Mapped Address, with Type 17 structures, Memory Device, temporary handle values will be provided to each Type 17 structure. Each Type 20 structure will have its MemoryDeviceHandle field set to the value of the Handle field of the associatied Type 17 structure. These temporary values allow the host environment to preserve the relationship of the two structure types in the host's SMBIOS table. The host SMBIOS table will use handle values calculated by the host environment, not the temporary handle values.*

AcpiPState

Pointer to the buffer area that contains the PState information. The buffer is allocated by using the Call-Out AgesaAllocateBuffer. The buffer will contain the PState information within an SSDT table structure.

AcpiSrat (deprecated)

This parameter is deprecated and will be removed in subsequent revisions.

AcpiSlit (deprecated)

This parameter is deprecated and will be removed in subsequent revisions.

AcpiWheaMce

Pointer to the buffer area that contains the WHEA-MCE sub-table. The buffer is allocated by using the Call-Out AgesaAllocateBuffer. The size of the table is contained within the WHEA standard definition table structure.

AcpiWheaCmc

Pointer to the buffer area that contains the WHEA-CMC sub-table. The buffer is allocated by using the Call-Out AgesaAllocateBuffer. The size of the table is contained within the WHEA standard definition table structure.

AcpiAlib

Pointer to the ACPI ASL library sub-table. The buffer is allocated by using the Call-Out AgesaAllocateBuffer. See "ACPI ASL Library" on page 332 for details.

AcpiIvrs

Pointer to the APCI IVRS sub-table. The buffer is allocated by using the Call-Out AgesaAllocateBuffer.

AcpiCrat

Pointer to the buffer area that contains the CRAT information. The buffer is allocated by using the Call-Out AgesaAllocateBuffer.

AcpiCdit

Pointer to the buffer area that contains the CDIT information. The buffer is allocated by using the Call-Out AgesaAllocateBuffer.

GnbLateConfiguration

A structure containing platform-specific operational characteristics. These characteristics are used only in this late time period

The ACPI and DMI structure definitions are defined by the industry standard to which they belong, except where noted. The tables are generated in a state ready for inclusion into their respective parent structures as defined by the standard. The Host Environment need only to link or copy them into the appropriate framework and no reformatting is required.

```
typedef struct {
     IN      BOOLEAN           DockedTdpHeadroom;
     IN      UINT8             GnbIoapicId;
     IN      UINT8             FchIoapicId;
} GNB_LATE_CONFIGURATION;
```

DockedTdpHeadroom

Motherboard designs that follow the AMD recommendations for mobile units will be able to take advantage of the extra heat dissipation capability of the dock. This control informs the thermal control software that the design conforms to the recommendations.

This value is initially declared by the build configuration element “BLDCFG_DOCKED_TDP_HEADROOM” on page 193. The value is presented here for possible host environment modification.

GnbIoApicId

The AGESA software will create an ACPI structure known as the I/O Virtualization Reporting Structure (IVRS). Part of this structure contains the IO APIC IDs for the GNB and FCH devices. These are assigned by the platform BIOS and set here for inclusion into the IVRS. This value is between 0x00 and 0xFF, inclusive. The deafult value assigned is 0xFF which will cause the IVRS entry for the device to be skipped.

FchIoApicId

IO APIC ID for the FCH device. See *GnbIoApicId* above.

Description

The main purpose of this function is to generate informational data tables used by the operating system. The individual tables can be selected for generation through the user selection entries on the input parameters.

This routine uses the Call-Out AgesaAllocateBuffer to allocate a buffer of the proper size to contain the data.

The code path separates the BSP from the APs and perform a separate and appropriate list of tasks for each class of core.

For the BSP, the following actions are performed:

– Allocate buffer space for the tables.
– Generate the table contents.
– Make sure that the CPU is in a known good power state before proceeding to boot the OS.

For the APs, the following actions are performed:

• Final register settings preparing for entry to OS.
• Establish the final PState for entry to OS.

Dependencies

This routine is expected to be executed late in the boot sequence after main memory has been initialized, after PCI enumeration has completed, after the host environment ACPI sub-system has started, after the host environment has taken control of the APs, but just before the start of OS boot.

The host environment must provide the required call-outs listed in “Required Call-Out Procedures” on page 146 to provide the buffer space in main memory and execute code on the APs. The host environment must register the created ACPI table in the main ACPI pointer tables. This may require moving the generated tables to another location in memory.

This procedure requires a stack. The host environment must establish the stack environment prior to making the call to this procedure. Some functions depend upon the preservation of the heap data across the shift from pre-memory environment to a post-memory environment. If that data was not preserved, then those functions cannot complete and an error is returned.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_ALERT	
AGESA_ERROR	The system could not allocate the needed amount of buffer space; or could not locate the artifact data block in memory. Likely cause: the host environment may not have preserved the data properly.

AmdS3Save (deprecated)

AmdInitRtb

This procedure is called very late in POST at the Ready-To-Boot (RTB) time. It performs last minute configuration and saves silicon component registers to the SMM save area in preparation to enter system suspend mode.

Both entry names will be supported for a time to maintain backward compatibility in present systems.

Prototype

```
AGESA_STATUS
AmdS3Save (
     IN OUT  AMD_S3SAVE_PARAMS       *AmdS3SaveParams
);
AGESA_STATUS
AmdInitRtb (
     IN OUT  AMD_RTB_PARAMS          *AmdInitRtbParams
);
```

Parameters

AmdS3SaveParams

AmdInitRtbParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
        OUT  AMD_S3_PARAMS           S3DataBlock;
     IN      FCH_INTERFACE           FchInterface;
} AMD_RTB_PARAMS;
```

S3DataBlock

Structure of pointers to storage locations where the processor context will be stored. This parameter is allowed to be null when the S3 support is removed.

PlatformConfig

A structure containing platform-specific operational characteristics. This is the same structure described in “AmdInitEarly” on page 103.

FchInterface

A structure containing FCH parameters. This is the same structure described in “AmdInitEnv” on page 124.

```
typedef struct {
      OUT   UINT32      Signature;
      OUT   UINT16      Version;
     IN OUT UINT32      Flags;
     IN OUT VOID        *NvStorage;
     IN OUT UINT32      NvStorageSize;
     IN OUT VOID        *VolatileStorage;
     IN OUT UINT32      VolatileStorageSize;
} AMD_S3_PARAMS;
```

Signature

"ASTR" for AMD Suspend-To-RAM

Version

S3 Params version number

Flags

Indicates operation specifics

NvStorage

Pointer to memory critical save state data. The data in this section must be available to the restore process before the main memory is re-started. Therefore it is targeted to be stored in non-volatile storage.

NvStorageSize

The size, in bytes, of the NvStorage region. If the size is returned as zero, the non-volatile storage should not be updated.

VolatileStorage

Pointer to remaining AMD save state data. The data in this section is not critical to re-starting the main memory. Therefore, it may be stored in the main memory; or in non-volatile storage as desired.

VolatileStorageSize

The size, in bytes, of the DRAM storage region.

Description

This procedure saves critical registers and/or configuration information for preservation across a system suspend mode. All actions needed to prepare the processor for suspend mode is performed, however this procedure does NOT initiate the suspend process. The host environment is expected to perform that duty.

These actions are performed by the BSP. The APs are not assigned any tasks at this time point.

The initializer routine will NULL out the save area pointers and sizes. This procedure will determine the size of storage needed for all the processor context, and make a call out to the environment for allocation of one buffer to store all of the data. Upon exit, the pointers and sizes within the `AMD_S3_PARAMS` structure will be updated with the appropriate addresses within the buffer that was allocated. The host environment is expected to then transfer the data pointed to by NvStorage to a non-volatile storage area, and the data pointed to by VolatileStorage to either a non-volatile storage area or system RAM that retains its content across suspend.

Dependencies

The host environment must initiate the suspend process.

This procedure requires a stack. The host environment must establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	All suspend duties have been completed successfully.
AGESA_ALERT	

14.3 Resume Branch Procedures

These routines are used to restore the processors to the operational state that was saved earlier in time.

AmdInitResume

This procedure performs silicon device and memory re-initialization for the resume boot path.

Prototype

```
AGESA_STATUS
AmdInitResume (
     IN      AMD_RESUME_PARAMS *ResumeParams
);
```

Parameters

ResumeParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      AMD_S3_PARAMS           S3DataBlock;
} AMD_RESUME_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics. This is the same structure described in “AmdInitEarly” on page 103.

S3DataBlock

Structure of pointers to storage locations where the processor context is stored. This is the same structure described in “AmdInitRtb” on page 137.

Description

This procedure initializes or re-initializes the silicon components for the resume boot path. For the processor, main memory is brought out of self-refresh mode. This procedure will use the context data in the NvStorage area of the input structure to re-start the main memory.

The host environment must fill the `AMD_S3_PARAMS` NvStorage and VolatileStorage pointers and related size elements to describe the location of the context data. Note for this procedure, the two data areas do not need to be contained in one buffer zone; they can be anywhere in the accessible memory address space. If the host environment uses a non-volatile storage device accessed on the system address bus such as flash ROM, then the context data does not need to be moved prior to this call. If the host environment uses a non-volatile storage device not

located on the system address bus (e.g. CMOS or SSEPROM) then the host environment must transfer the context data to a buffer in main memory prior to calling this procedure.

These actions are performed by the BSP. The APs are not assigned any tasks at this time point.

Dependencies

The host environment must have determined that the system should take the resume path prior to calling this procedure. The configuration done by AmdInitEarly and any necessary warm reset must be complete. After this procedure, execution proceeds to general system restoration.

The host environment must restore MTRRs.

This procedure requires a stack. The host environment must use one of the provided service functions to establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	Re-initialization has been completed successfully.
AGESA_ALERT	
AGESA_ERROR	

AmdS3LateRestore

This procedure restores saved registers after the PCI control structures have been re-established.

Prototype

```
AGESA_STATUS
AmdS3LateRestore (
     IN OUT  AMD_S3LATE_PARAMS *S3LateParams
);
```

Parameters

S3LateParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      AMD_S3_PARAMS           S3DataBlock;
} AMD_S3LATE_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics. This is the same structure described in “AmdInitEarly” on page 103.

S3DataBlock

Structure of pointers to storage locations where the processor context is stored. This is the same structure described in “AmdS3Save (deprecated)” on page 137.

Description

This procedure restores the processor state, reloads critical silicon component registers, and performs any re-initialization required by the silicon. This procedure will use the context data in the VolatileStorage area of the input structure to restore the processor registers.

The host environment must fill the `AMD_S3_PARAMS` NvStorage and VolatileStorage pointers and related size elements to describe the location of the context data. Note for this procedure, the two data areas do not need to be contained in one buffer zone; they can be anywhere in the accessible memory address space. If the host environment uses a non-volatile storage device accessed on the system address bus such as flash ROM, then the context data does not need to be moved prior to this call. If the host environment uses a non-volatile storage device not located on the system address bus (e.g. CMOS or SSEPROM) then the host environment must transfer the context data to a buffer in main memory prior to calling this procedure.

These actions are performed by the BSP. The APs are not assigned any tasks at this time point.

Dependencies

This procedure is called late in the resume sequence, after the PCI control space is restored.

The host environment must initiate the OS restart process.

This procedure requires a stack. The host environment must establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	All resume processes have been completed successfully.
AGESA_ALERT	
AGESA_WARNING	

AmdS3FinalRestore

This procedure restores saved registers and prepares the silicon components for OS restart.

Prototype

```
AGESA_STATUS
AmdS3FinalRestore (
     IN OUT  AMD_S3FINAL_PARAMS*S3LateParams
);
```

Parameters

S3LateParams

Pointer to the parameter structure described below.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS StdHeader;
     IN      PLATFORM_CONFIGURATION  PlatformConfig;
     IN      AMD_S3_PARAMS           S3DataBlock;
} AMD_S3FINAL_PARAMS;
```

PlatformConfig

A structure containing platform-specific operational characteristics. This is the same structure described in “AmdInitEarly” on page 103.

S3DataBlock

Structure of pointers to storage locations where the processor context is stored. This is the same structure described in “AmdInitRtb” on page 137.

Description

This procedure performs any last restoration or re-initialization required by the silicon prior to OS re-entry. This entry point is called at the very end of the resume process, just prior to OS handoff, after the PEI_POST_BOOT_SCRIPT_TABLE_PPI_GUID is installed.

Dependencies

This procedure is called last in the resume sequence, just before resuming operating system execution.

The host environment must initiate the OS restart process.

This procedure requires a stack. The host environment must establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS	All resume processes have been completed successfully.

Chapter 15 Call-Out Procedures

15.1 Required Call-Out Procedures

The procedures listed in this section are required to be implemented by the host environment. These functions provide essential information without which the AGESA™ software cannot function properly. Optional call-out functions are described in later sections along with the entry procedure that uses the specific call-out.

AgesaAllocateBuffer CallOut

This host environment function is used in the post memory period to request the host environment use its internal methodology to allocate a section of main memory for use as a data buffer. **required**

Prototype

```
AGESA_STATUS
AgesaAllocateBuffer (
     IN      UINTN                   FcnData,
     IN OUT  AGESA_BUFFER_PARAMS     *AllocParams
);
```

Parameters

FcnData

This standard call-out parameter is not used for this call-out.

AllocParams

Pointer to a structure containing the requirements for the requested memory block.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     IN OUT  UINT32                  BufferLength;
     IN      UINT32                  BufferHandle;
     OUT     VOID                    *BufferPointer;
} AGESA_BUFFER_PARAMS;
```

StdHeader

AMD standard Header structure. See “Standard Header” on page 32.

BufferLength

On input, this is the requested size to be allocated. Upon successful completion, this is updated by the call-out procedure to indicate the amount actually allocated.

BufferHandle

This is an identifier used to name and locate the allocated block at some future time. The AGESA™ software uses the buffer handles defined in the enum list AMD_BUFFER_HANDLE for the buffers it needs to allocate. The host environment software may use BufferHandle to apply special processing to certain buffer types.

BufferPointer

This is a pointer to the memory space allocated by the host environment for use as the data buffer. Upon successful completion, this is updated by the call-out procedure with a 32-bit mode address of the buffer. The pointer should be NULL if allocation was not successful.

Description

This function is used after main memory has been initialized and the host environment has taken control of memory allocation. This function must allocate a buffer of the requested size or larger. This function is **required** to be implemented by the host environment.

Beginning at the AmdInitEnv boot time point and continuing, the item `StdHeader->HeapStatus` should be checked by the callout implementation for indicating HEAP_RUNTIME_SYSTEM_MEM, rather than a boot time point. When HEAP_RUNTIME_SYSTEM_MEM is indicated, the callout implementation should cause the allocated memory to persist at run-time as a reserved address region. For UEFI, the callout implementation can map HEAP_RUNTIME_SYSTEM_MEM to EfiACPIMemoryNVS or EfiRuntimeServicesData.

Dependencies

The following call-outs must work together in the host system. Parameters of the same name have the same function and must be treated the same in each function:

AgesaAllocateBuffer	AgesaDeallocateBuffer
AgesaLocateBuffer	AgesaRunFcnOnAp

The host environment may need to reserve a location in the buffer to store any host environment specific value(s). The returned pointer must not include this reserved space. The host environment on the AgesaDeallocateBuffer call needs to account for the reserved space. This reserved space may be an identifier or the “handle” used to identify the specific memory block.

Status Codes Returned

AGESA_SUCCESS	The requested size of memory has been successfully allocated.
AGESA_UNSUPPORTED	This is a required function, so this value being returned causes a critical error response value from the AGESA™ software function.
AGESA_WARNING	Less than the requested amount of memory was allocated.

AgesaDeallocateBuffer CallOut

This host environment function de-allocates a valid buffer that was previously obtained from the AgesaAllocateBuffer function. **Required**

Prototype

```
AGESA_STATUS
AgesaDeallocateBuffer
(
     IN      UINTN       FcnData,
     IN OUT  AGESA_BUFFER_PARAMS     *DeallocParams
);
```

Parameters

FcnData

This standard call-out parameter is not used for this call-out.

DeallocParams

This is the same structure used by the AgesaAllocateBuffer function.

Related Definitions

AGESA_BUFFER_PARAMS

Note that this structure is the same as that used for AgesaAllocateBuffer. Please refer to that function definition for parameter details. In this usage, only the standard header and the BufferHandle elements need to be populated by the caller. This is sufficient to identify the buffer.

Description

This function is used after main memory has been initialized and the host environment has taken control of memory allocation. This function releases a valid working buffer. This function is **required** for the host environment to implement.

Dependencies

The following call-outs must work together in the host system. Parameters of the same name have the same function and must be treated the same in each function:

AgesaAllocateBuffer	AgesaDeallocateBuffer
AgesaLocateBuffer	AgesaRunFcnOnAp

Status Codes Returned

The function must return a status value indicating one of the following:

AGESA_SUCCESS The function has completed successfully.

AGESA_BOUNDS_CHK The BufferHandle is invalid. The AGESA™ software continues with its function.

AGESA_UNSUPPORTED This is a required function, so this value being returned causes a critical error response value from the AGESA™ software function.

AgesaDoReset CallOut

This function does a system reset. **Required**

Prototype

```
VOID
AgesaDoReset (
     IN      UINTN                   ResetType,
     IN OUT  AMD_CONFIG_PARAMS       *StdHeader
);
```

Parameters

ResetType

This standard call-out parameter indicates which type of system reset to initiate; either a “cold” reset or a “warm” reset.

StdHeader

Pointer to a data structure containing the AMD standard Header structure.

Related Definitions

```
#define     WARM_RESET_WHENEVER     1
#define     COLD_RESET_WHENEVER     2
#define     WARM_RESET_IMMEDIATELY  3
#define     COLD_RESET_IMMEDIATELY  4
```

A cold reset is characterized by the momentary deassertion of the PWRGOOD/PWROK signal. A warm reset is characterized by the momentary assertion of the RESET# signal with no change to the PWRGOOD/PWROK signal.

Description

This host environment function must initiate the specified type of system reset.

Implementation of this function by the host environment is REQUIRED. Some host environments may record this as a request allowing other elements in the system to perform some additional tasks before the actual reset is issued.

Dependencies

The AMD processor contains 3 bits (BiosRstDet[2:0]) in a PCI register (F0x6C Link Initialization Control Register) that indicate the reset status. These bits are reserved for use by the AGESA™ software and should not be modified by the host environment.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AGESA_UNSUPPORTED This is a required function, so this value being returned causes a critical error response value from the AGESA™ software function.

AgesaLocateBuffer CallOut

This host environment function returns a pointer to the buffer with the indicated handle. **Required**

Prototype

```
AGESA_STATUS
AgesaLocateBuffer
(
     IN      UINTN                   FcnData,
     IN OUT  AGESA_BUFFER_PARAMS     *LocateParams
);
```

Parameters

FcnData

This standard call-out parameter is not used for this call-out.

LocateParams

This is the same structure used by the AgesaAllocateBuffer function.

Related Definitions

`AGESA_BUFFER_PARAMS`

Note that this structure is the same as that used for AgesaAllocateBuffer. Please refer to that function definition for parameter details. In this usage, only the standard header and the BufferHandle elements need to be populated by the caller. This is sufficient to identify the buffer. The implemented routine must locate the buffer and then populate BufferLength and BufferPointer. If the requested buffer is not found, BufferLength should be zero and BufferPointer should be NULL.

Description

This function is used after main memory has been initialized and the host environment has taken control of memory allocation. This function must locate the buffer related to the indicated handle and return the address of the buffer and its length. This function is **required** to be implemented in the host environment.

Dependencies

The following call-outs must work together in the host system. Parameters of the same name have the same function and must be treated the same in each function:

AgesaAllocateBuffer	AgesaDeallocateBuffer
AgesaLocateBuffer	AgesaRunFcnOnAp

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_BOUNDS_CHK	The presented handle is invalid or the buffer could not be located.

AgesaReadSpd CallOut

AgesaReadSpdRecovery (Deprecated) CallOut

These call-outs read a block of memory SPD data and places it into the provided buffer. **required**

Prototype

```
AGESA_STATUS
AgesaReadSpd (
     IN      UINTN                    FcnData,
     IN OUT  AGESA_READ_SPD_PARAMS    *ReadSpd
);
```

Parameters

FcnData

This standard call-out parameter is not used for this call-out.

ReadSpd

Pointer to a data structure containing the parameter information.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS StdHeader;
     IN      UINT8             SocketId;
     IN      UINT8             MemChannelId;
     IN      UINT8             DimmId;
     IN OUT  UINT8             *Buffer;
     IN OUT  MEM_DATA_STRUCT   *MemData;
} AGESA_READ_SPD_PARAMS;
```

StdHeader

AMD standard Header structure. See “Standard Header” on page 32.

SocketID

Identifier indicating the socket where the target SPD is located. The motherboard designer must follow the design guidelines to make this match the silk screen labels on the board. Otherwise, the host environment needs to provide their own translation procedures.

MemChannelId

Index of the memory channel for which to read the SPD data. Channels are numbered 0 to 3 relative to the socket. Channel 0 matches with “channel A” in the processor specification, channel 1 with “channel B,”... channel 3 with “channel D.”

DimmID

Index of the DIMM for which to read the SPD data. DIMMs are numbered 0 to 3 relative to the channel. Each channel has memory chip-select lines that define the DimmID:

Table 1: DIMM number identification

	Channel A	Channel B	Channel C	Channel D
DIMM 0	MA0_CS_L[0,1]	MB0_CS_L[0,1]	MC0_CS_L[0,1]	MD0_CS_L[0,1]
DIMM 1	MA1_CS_L[0,1]	MB1_CS_L[0,1]	MC1_CS_L[0,1]	MD1_CS_L[0,1]
DIMM 2	MA2_CS_L[0,1]	MB2_CS_L[0,1]	MC2_CS_L[0,1]	MD2_CS_L[0,1]
DIMM 3	MA3_CS_L[0,1]	MB3_CS_L[0,1]	MC3_CS_L[0,1]	MD3_CS_L[0,1]

Buffer

Pointer to a location to store the SPD data read from the DIMM. All 256 bytes of the DIM SPD should be read from the device and stored in the buffer. Use of the Block Read protocol is recommended for speed of operation.

Description

This procedure is called by the host environment call-out router. This call-out is used before main memory is initialized.

The SPD data is read over the SMBus using a platform specific SMBus controller. If supported by the SMBus controller, a block read operation is recommended to reduce the transfer time. The host environment must adjust any SMBus routing required to access the target DIMM.

Dependencies

The host environment is ***required*** to provide this function. The main system memory cannot be properly initialized without the SPD data.

Status Codes Returned

AGESA_SUCCESS Indicates the SPD block for the indicated DIMM was read successfully.

AGESA_BOUNDS_CHK The specified DIMM is not present.

AGESA_UNSUPPORTED This is a required function, so this value being returned causes a critical error response value from the AGESA™ software function and no memory initialized.

AGESA_ERROR The DIMM SPD read process has generated communication errors.

AgesaRunFcnOnAp CallOut

This function causes code to be executed on an Application Processor. **Required**

This function is tightly tied to the following entry point AmdLateRunApTask.

Prototype

```
AGESA_STATUS
AgesaRunFcnOnAp (
     IN      UINTN              ApicIdOfCore,
     IN      AP_EXE_PARAMS      *LaunchApParams
);
```

Parameters

ApicIdOfCore

This is the APIC ID of the target AP that is to execute the function.

LaunchApParams

Pointer to a structure containing the AP address and function data.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS StdHeader;
     IN      UINT32            FunctionNumber;
     IN      VOID              *RelatedDataBlock;
     IN      UINT32            RelatedBlockLength;
} AP_EXE_PARAMS;
```

FunctionNumber

This is a parameter to the AP that indicates the sub-function to perform. The AGESA™ software AP dispatcher uses this function number to select the target procedure.

RelatedDataBlock

This is a pointer to a data block related to the function that the AP must perform. The host environment must make this data block available to the AP during its execution of the specified function. Simultaneous access to the data block by multiple processors is **not** required.

RelatedBlockLength

Length of the data block related to the function the AP must perform. If the length is zero, then there is no related data block.

Description

This function is used after main memory has been initialized and the host environment has taken control of AP task dispatching. This function must cause the indicated function code to be executed upon the specified Application Processor. This procedure must be executed in 32-bit mode. This function is **required** to be implemented in the host environment.

Dependencies

The host environment must route execution to the target AP and have that AP call the“AmdLateRunApTask” entry point.

Status Codes Returned

The status code returned to the caller must also reflect the status returned by the AP when executing the requested function.

AGESA_SUCCESS	The function has completed successfully.
AGESA_UNSUPPORTED	This is a required function, so this value being returned causes a critical error response value from the AGESA™ software function and no memory initialized.
AGESA_WARNING	The AP did not respond.

AmdLateRunApTask CallOut

This procedure is run by the Application Processor to perform a function as directed by the BSP.

Prototype

```
AGESA_STATUS
AmdLateRunApTask (
     IN AP_EXE_PARAMS  *AmdApExeParams
);
```

Parameters

AmdApExeParams

Pointer to the interface structure for requested routine.

Related Definitions

AP_EXE_PARAMS

This is the same structure used for “AgesaRunFcnOnAp” on page 157.

Description

This entry point is tightly connected with the “AgesaRunFcnOnAp” on page 157. The AGESA™ software will call the call-out “AgesaRunFcnOnAp”; the host environment will then call this entry point to have the AP execute the requested function. This is needed late in the Post and Resume branches for running an AP task since the AGESA™ software has relinquished control of the APs to the host environment.

Dependencies

The host environment must implement the“AgesaRunFcnOnAp” call-out and route execution to the target AP.

Status Codes Returned

The status code returned by this procedure must also be returned to the original caller of “AgesaRunFcnOnAp” on page 157.

AGESA_SUCCESS	The function has completed successfully.
AGESA_UNSUPPORTED	The requested function is unknown.
AGESA_BOUNDS_CHECK	The MMIO base is not set.

AgesaPcieSlotResetControl CallOut

This function does a PCIe port reset.

Prototype

```
AGESA_STATUS
AgesaPcieSlotResetControl (
  IN      UINTN                 FcnData,
  IN      PCIe_SLOT_RESET_INFO  *ResetInfo
  );
```

Parameters

ResetInfo

Pointer to the interface structure for requested routine.

Related Definitions

PCIe_SLOT_RESET_INFO

```
typedef struct {
  IN      AMD_CONFIG_PARAMS     StdHeader;
  IN      UINT8                 ResetId;
  IN      UINT8                 ResetControl;
} PCIe_SLOT_RESET_INFO;
```

StdHeader

The current configuration.

ResetId

The port, or group of ports, to be reset. Often corresponds to a GPIO.

ResetControl

Specifies the requested reset behavior. This is specified using PCIE_RESET_CONTROL enum.

AssertSlotReset - Assert the reset.
DeassertSlotReset - Deassert the reset.

Description

This entry point performs PCIe port resets, by knowing how the platform maps each reset ID.

Dependencies

The reset ID for the port is provided as part of the PCIe topology.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_UNSUPPORTED	The requested function is unknown.

15.2 Optional Call-Out Procedures

AgesaGetIdsData CallOut

Read a portion of IDS data from the non-volatile (NV) storage. **Optional**

Prototype

```
AGESA_STATUS
AgesaGetIdsData (
     IN      UINTN                   FcnData,
     IN OUT  IDS_CALLOUT_STRUCT      *IdsCalloutData
);
```

Parameters

FcnData

This parameter indicates the IDS sub-function desired.

IdsCalloutData

Pointer to a structure containing requirements for the IDS function

Related Definitions

```
typedef struct {
     IN      AMD_CONFIG_PARAMS       StdHeader;
     IN OUT  IDS_NV_ITEM             *IdsNvPtr;
     IN OUT  UINTN                   Reserved;
} IDS_CALLOUT_STRUCT;
```

StdHeader

AMD standard Header structure. See “Standard Header” on page 32.

IdsNvPtr

Pointer to a buffer where the host environment should place the requested data.The data is an array of ID-Value pairs. The array is terminated with an entry containing 0xffff in the IdsNvId field.

```
typedef struct {
     IN      UINT16            IdsNvId;
      OUT    UINT16            IdsNvValue;
} IDS_NV_ITEM;
```

IdsNvId

This is the unique data element identifier.

IdsNvValue

This is the value of the data element.

Description

This host environment procedure must walk the array list of IDS element identifiers, translate the identifier to a platform NV storage location, read the value from the NV storage, and store that value into the array's value entry. The identifiers for the IDS data elements are defined in the IDS.H include file.

This call-out is listed as optional since the IDS sub-system is also optional. However, if the IDS features are included into the build, then this call-out becomes required.

Dependencies

The NV storage must contain valid data values for the IDS elements.

Status Codes Returned

AGESA_SUCCESS	The host environment has successfully fill the array of IDS_NV_ITEM.
AGESA_ALERT	The IDS_NV_ITEM list contained an unrecognized IdsNvId.
AGESA_UNSUPPORTED	The host environment returning this value causes the default value(s) to be used for the related data elements.

AgesaHeapRebase CallOut

Allow the host environment to control the initial heap location.

Prototype

```
AGESA_STATUS
AgesaLocateBuffer
(
     IN      UINTN                   FcnData,
     IN OUT  AGESA_REBASE_PARAMS     *RebaseParams
);
```

Parameters

FcnData

This standard call-out parameter is not used for this call-out.

RebaseParams

This structure communicates the heap base address.

Related Definitions

```
typedef struct {
     IN OUT  AMD_CONFIG_PARAMS       StdHeader;
     OUT     UINTN                   *HeapAddress;
} AGESA_BUFFER_PARAMS;
```

StdHeader

AMD standard Header structure. See “Standard Header” on page 32.

HeapAddress

Provides the base address to use for the heap data.

Description

Implementation of this function is **optional** for the host environment. This call-out is an opportunity for the host environment provide the heap base address to be used. This function can be used to provide alternate heap locations when the system is performing a secure resume and the default location is not available. If the function is not implemented or does not update the HeapAddress parameter, a default location will be used for the heap data.

Dependencies

None.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.

AgesaHookBeforeDramInit CallOut

General-purpose hook called before the DRAM_Init bit is set. **Optional**

Prototype

```
AGESA_STATUS
AgesaHookBeforeDramInit (
     IN      UINTN             SocketIdModuleId,
     IN OUT  MEM_DATA_STRUCT   *MemData
);
```

Parameters

SocketIdModuleId

This standard call-out parameter provides the processor socket and internal node number for the MCT which is about to be enabled. These bitfields are accessed as shown:

SocketIdModuleId.SocketId - The processor's socket.
SocketIdModuleId.ModuleId - The internal node for this MCT.

MemData

Pointer to a data structure containing the memory information. This is the same data structure that was passed to the main memory routine. See "AmdInitPost" on page 114 for structure details.

Description

General-purpose hook called before the DRAM_Init bit is set. Called once per MCT.

Implementation of this function is **optional** for the host environment. This call-out is an opportunity for the host environment to make dynamic modifications to the memory timing settings specific to the board or host environment.

New structure member to note is `VddpVddrVoltage`. If the `IsValid` element is TRUE, the platform is expected to change the VRM controls to modify the voltage supplied to the Memory PHY power pin to the voltage specified. Please check the Power specification for the APU. Specific power requirements for the APU may change the implementation status of this hook to Required. This is presently the case for the Kaveri SoCs.

Dependencies

Status Codes Returned

AGESA_SUCCESS The function has completed successfully.

AGESA_UNSUPPORTED This function is not implemented by the host environment.

AGESA_WARNING

AgesaHookBeforeDramInitRecovery (Deprecated) CallOut

AgesaHookBeforeDQSTraining CallOut

General-purpose hook called before the memory training begins. **Optional**

Prototype

```
AGESA_STATUS
AgesaHookBeforeDQSTraining (
     IN      UINTN               SocketIdModuleId,
     IN OUT  MEM_DATA_STRUCT    *MemData
);
```

Parameters

SocketIdModuleId

This standard call-out parameter provides the processor socket and internal node number for the MCT which is about to be enabled. These bitfields are accessed as shown:

SocketIdModuleId.SocketId - The processor's socket.
SocketIdModuleId.ModuleId - The internal node for this MCT.

MemData

Pointer to a data structure containing the memory information. This is the same data structure that was passed to the main memory routine. See "AmdInitPost" on page 114 for structure details.

Description

General-purpose hook called just before the memory training processes begin. Called once per MCT.

Implementation of this function is **optional** for the host environment. This call-out is an opportunity for the host environment to make dynamic modifications to the memory timing settings specific to the board or host environment.

The host environment may also use this call-out for some board-specific features that should be activated at this time point, such as:

- Low voltage DIMMs—the host environment should set the recommended voltages found in the memory data structure for each memory channel. This needs to occur before training begins.

Status Codes Returned

AGESA_SUCCESS The function has completed successfully.

AGESA_UNSUPPORTED This function is not implemented by the host environment.

AgesaExternal2dTrainVrefChange CallOut

Called in order to change the external Vref for 2D memory training. **Optional**

Prototype

```
AGESA_STATUS
AgesaExternal2dTrainVrefChange (
     IN      UINTN              SocketIdModuleId,
     IN OUT  MEM_DATA_STRUCT   *MemData
);
```

Parameters

SocketIdModuleId

This standard call-out parameter provides the processor socket and internal node number for the MCT which is about to be enabled. These bitfields are accessed as shown:

SocketIdModuleId.SocketId - The processor's socket.
SocketIdModuleId.ModuleId - The internal node for this MCT.

MemData

Pointer to a data structure containing the memory information. This is the same data structure that was passed to the main memory routine. See "AmdInitPost" on page 114 for structure details.

Description

When indicated, this call out will be called in order to change the external Vref rather than using internal Vref control. The host environment should implement the platform specific external Vref control.

Dependencies

Status Codes Returned

AGESA_SUCCESS The function has completed successfully.

AGESA_UNSUPPORTED This function is not implemented by the host environment.

AgesaHookBeforeExitSelfRefresh CallOut

General-purpose hook called before exiting self-refresh mode during the resume process. **Optional**

Prototype

```
AGESA_STATUS
AgesaHookBeforeExitSelfRefresh (
     IN      UINTN                           FcnData,
     IN OUT  MEM_DATA_STRUCT                 *MemData
);
```

Parameters

FcnData

This standard call-out parameter is not used for this call-out.

MemData

Pointer to a data structure containing the memory information. This is the same data structure that was passed to the main memory routine. See “AmdInitPost” on page 114 for structure details.

Description

General purpose hook called before the exiting self refresh. This procedure is called once per channel.

Implementation of this function is **optional** for the host environment. This call-out is an opportunity for the host environment to make dynamic modifications to the memory timing settings specific to the board or host environment before exiting self refresh on S3 resume.

Dependencies

This procedure is called before the exit self refresh bit is set in the resume sequence. The host environment must initiate the OS restart process. This procedure requires a stack. The host environment must establish the stack environment prior to making the call to this procedure.

Status Codes Returned

AGESA_SUCCESS The function has completed successfully.

AGESA_UNSUPPORTED This function is not implemented by the host environment.

AGESA_WARNING

AgesaGetVbiosImage CallOut

This function does a PCIe port reset.

Prototype

```
AGESA_STATUS
AgesaGetVbiosImage (
  IN        UINTN                 FcnData,
  IN OUT    GFX_VBIOS_IMAGE_INFO *VbiosImageInfo
  );
```

Parameters

VbiosImageInfo

Pointer to the video BIOS and control information.

Related Definitions

```
typedef struct {
  IN      AMD_CONFIG_PARAMS     StdHeader;
  OUT     VOID                  *ImagePtr;
  IN      PCI_ADDR              GfxPciAddress;
  IN      UINT32                Flags;
} GFX_VBIOS_IMAGE_INFO;
```

StdHeader

The current configuration.

ImagePtr

A pointer to the video BIOS image.

GfxPciAddress

The configuration base address of the integrated graphics controller.

Flags

Indicate whether a special repost is required.

Description

This entry point performs the special video BIOS repost call, if indicated, and returns a pointer to the video BIOS image.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_UNSUPPORTED	The requested function is unknown.

AgesaFchOemCallout CallOut

This function provides expert level FCH customization.

Prototype

```
AGESA_STATUS
AgesaGetVbiosImage (
  IN        UINTN       FcnData,
  IN OUT    VOID        *FchData
  );
```

Parameters

FchData

Pointer to the current FCH data block structure.

Related Definitions

During PEI phase this points to FCH_RESET_DATA_BLOCK and during DXE phase this points to FCH_DATA_BLOCK. Refer to the internal documentation for the latest and most detailed information on these two structures (“Internal Documentation” on page 25).

Description

This callout provides the ability to make expert level customizations prior to both the PEI and DXE FCH initialization.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_UNSUPPORTED	The requested function is unknown.

AgesaExternalVoltageAdjust CallOut

Called in order to change the external voltage as part of the memory test feature. Optional

While optional in general, this callout is required when using the MBIST feature.

Prototype

```
AGESA_STATUS
AgesaExternalVoltageAdjust (
     IN      UINTN              SocketIdModuleId,
     IN OUT  VOLTAGE_ADJUST    *AdjustValue
);
```

Parameters

SocketIdModuleId

This standard call-out parameter provides the processor socket and internal node number for the MCT which is about to be enabled. These bitfields are accessed as shown:

SocketIdModuleId.SocketId - The processor's socket.
SocketIdModuleId.ModuleId - The internal node for this MCT.

AdjustValue

Pointer to a data structure containing the voltage adjustment information.

Related Definitions

VOLTAGE_ADJUST

```
typedef struct {
        OUT  AMD_CONFIG_PARARMS      StdHeader;
        OUT  MEM_DATA_STRUCT         *MemData;
     IN      UINT8                   VoltageType;
     IN      INT8                    AdjustValue;
} VOLTAGE_ADJUST;
```

StdHeader

The current configuration.

MemData

Provides access to current memory data.

VoltageType

Specifies the type of voltage adjustment to make.

CPU_VREF - (Optional.) Normally this adjustment is made using processor hardware support. If desired by the platform, this callout can be used to

accomplish this adjustment with platform specific hardware support. To enable callout control provide the following build option:
`#define CFG_ENABLE_EXTERNAL_VREF TRUE`

DIMM_VREF - A request to vary the DIMM side Vref voltage level in increments of +/- 1% from the level determined on the platform.

VDDIO - A request to vary the VDDIO voltage in increments of +/-1% from the nominal VDDIO set by the platform.

AdjustValue

The signed adjustment value to be made to the voltage, in percentage.

Description

When the memory testing feature is enabled, this callout will be used to control voltage levels in order to stress test memory and verify the integrity of the memory subsystem.

Dependencies

Code is running from Cache-As-RAM (CAR) environment. Memory testing is destructive to memory content.

Status Codes Returned

AGESA_SUCCESS The function has completed successfully.

AGESA_UNSUPPORTED This function is not implemented by the host environment.

AgesaGnbOemCallout CallOut

Called to request a change to some platform specific controls. Optional

While optional in general, this callout is required when the host platform implements specific hardware for tuning performance.

Prototype

```
AGESA_STATUS
AgesaGnbOemCallout (
     IN      UINTN              FcnData,
     IN      PCI_ADDR          *Device
);
```

Parameters

FcnData

This standard call-out parameter is which specifies the function to be performed.

Device

This is a pointer to a PCI address for the target device.

Related Definitions

FcnData == AGESA_GNB_PCIE_CLK_REQ

The PCIe device, addressed by *Device, is capable of supporting the “CLKREQ#” mode of operation. This call-out is requesting the host environment to configure the clock generator device to also operate in the CLKREQ# mode. Upon succesful return, the PCIe device will

Description

This is a general purpose call-out that requests the host environment to perform certain actions upon platform specific devices outside the control of the AGESA code. The requested action is indicated by the *FcnData* parameter.

Status Codes Returned

AGESA_SUCCESS The function has completed successfully.

AGESA_UNSUPPORTED This function is not implemented by the host environment.

Chapter 16 Customizing Platforms

Some platforms require special handling of certain characteristics. The following sections describe the customization controls available. Many of these controls require an advanced knowledge level and should not be used until the platform developer fully understands the processes involved.

There are four mechanisms available to the host environment to provide customized data:

1. <Plat>Options.c Build Configuration settings—user settings are defined in below. They fall into two categories:

   a. Platform-specific limits—predefined configuration elements to describe the platform

   b. User-defined register settings—user-defined register setting table.

   This form of customization is used for items that change from platform to platform but remain static on any single platform, meaning they don't change from boot to boot nor are they likely to be the subject of a end-user choice.

2. Wrapper interaction after the call to AmdCreateStruct — as discussed in "Call Mechanics" on page 26, the host environment has an opportunity to modify parameters and/or register values after the call to AmdCreateStruct completes. This is a perfect place to insert user-specified settings retrieved from non-volatile storage (for example, host BIOS SETUP) or to make modifications based on a programmed algorithm.

   This form of customization is used for items that differ from the AMD defaults and are the same for a large sub-class of platforms, or are subject to an end-user choice not covered by the IDS features. For example, an IBV or OEM may use this form of customization to establish a corporate standard.

3. Integrated Debug Services (IDS) controls — these are described in "IDS Configuration Controls" on page 246. They are only available during the platform 'bring-up' and debug phase.

4. Defined call-outs — listed in "Optional Call-Out Procedures" on page 162, these call-outs occur at specific points in the initialization sequence. The host environment can perform register modifications and/or algorithmic modifications at these times.

   This form of customization is used for items that require active modification at a very specific point inside a running algorithm. Execution control is given over to a host environment sub-routine.

16.1 Build Options

Build options are boolean constants. The purpose of build options is to remove code from the build to reduce the overall code size present in the ROM image. Unless otherwise specified, the default action is to include all options. If a build option is not specifically listed as disabled, then it is included into the build.

The user specifies which build options are desired in a source file local to their project named <plat>Options.c This file is compiled with the AGESA™ project, therefore the build control file needs to know the location of the platform project. The user must set the environment variable AGESA_OptsDir to the path to find the <plat>Options.c file.

To disable an option, a line such as the following must be included in the Options.c file:

```
#define     BLDOPT_REMOVE_???_SUPPORT     TRUE
```

A value of TRUE disables or removes the option and the associated code is removed from the build. A value of FALSE means that the option is enabled and included into the build. A build option does not need to be listed in the Options.c file unless it is being changed from its default value. The defined build options are:

BLDOPT_REMOVE_UDIMMS_SUPPORT

If unbuffered DIMMs are NOT expected to be required in the system, the code that handles unbuffered DIMMs can be removed from the build.

BLDOPT_REMOVE_RDIMMS_SUPPORT

If registered DIMMs are NOT expected to be required in the system, the code that handles registered DIMMs can be removed from the build.

BLDOPT_REMOVE_LRDIMMS_SUPPORT

If Load Reduced (LR) DIMMs are NOT expected to be required in the system, the code that handles LR DIMMs can be removed from the build.

Note: *The above three options operate independently from each other; however, at least one of the unbuffered, registered, or load reduced options must be present in the build.*

BLDOPT_REMOVE_DDR3_SUPPORT

For systems supporting DDR3 memory and another memory type, such as GDDR5, the code for DDR3 may be removed if DDR3 memory is NOT expected to be required in the system. If the processor only supports DDR3 memory, this option is ignored.

BLDOPT_REMOVE_GDDR5_SUPPORT

For systems supporting GDDR5 memory and another memory type, such as DDR3, the code for GDDR5 may be removed if GDDR5 memory is NOT expected to be required in the system. If the processor only supports GDDR5 memory, this option is ignored.

BLDOPT_REMOVE_ECC_SUPPORT

Use this option to remove the code for Error Checking & Correction.

BLDOPT_REMOVE_DCT_INTERLEAVE

Interleaving is a mechanism to do performance fine tuning. This option interleaves memory from two DRAM controllers.

BLDOPT_REMOVE_BANK_INTERLEAVE

Interleaving is a mechanism to do performance fine tuning. This option interleaves memory between banks on a DIMM.

BLDOPT_REMOVE_NODE_INTERLEAVE

Interleaving is a mechanism to do performance fine tuning. This option interleaves memory from two HyperTransport™ nodes.

BLDOPT_REMOVE_PARALLEL_TRAINING

For multi-socket systems, training memory in parallel can reduce the time needed to boot.

BLDOPT_REMOVE_HW_RDDQS_2D_TRAINING

Memory training that can take voltage into account as well as timing, is 2D training.

BLDOPT_REMOVE_ONLINE_SPARE_SUPPORT

Online Spare support is removed by this option. The operation of this feature is described in “On-Line Spares” on page 277.

BLDOPT_REMOVE_ON_DIMM_THERMAL_SUPPORT

Support for on DIMM thermal sensors is removed by this option.

BLDOPT_REMOVE_AMP_SUPPORT

Support for the AMP memory overclocking feature is removed by this option.

BLDOPT_REMOVE_MULTISOCKET_SUPPORT

Many systems use only a single socket and may benefit in code space to remove this code. However, certain processors have multiple HyperTransport™ nodes within a single socket. For these processors, the multi-node support is required and this option has no effect.

BLDOPT_REMOVE_ACPI_PSTATES

This option removes the code that generates the ACPI tables used in power management.

BLDOPT_REMOVE_SRAT (deprecated)

This option removes the code that generates the SRAT tables used in performance tuning.

BLDOPT_REMOVE_SLIT (deprecated)

This option removes the code that generates the SLIT tables used in performance tuning.

BLDOPT_REMOVE_WHEA

This option removes the code that generates the WHEA tables used in error handling and reporting.

BLDOPT_REMOVE_DMI

This option removes the code that generates the DMI tables used in system management.

BLDOPT_REMOVE_DQS_TRAINING

This option removes the code used in memory performance tuning.

BLDOPT_REMOVE_C6_STATE

This option removes the code which implements the C6 C-State feature.

BLDOPT_REMOVE_LOW_POWER_STATE_FOR_PROCHOT

This option removes the code which implements adding a lower power P-state for PROCHOT conditions, for processors which have this support documented. For processors which do not include this support, this option has no effect.

BLDOPT_REMOVE_CRAT

This option removes the code which implements the ACPI Component Resource Affinity Table (CRAT).

BLDOPT_REMOVE_CDIT

This option removes the code which implements the ACPI Component Locality Distance Information Table (CDIT).

BLDOPT_REMOVE_MEM_RESTORE_SUPPORT

This option removes the memory context restore feature.

BLDOPT_REMOVE_EARLY_SAMPLES

Special support for Early Samples is included. Default setting is FALSE.

BLDOPT_REMOVE_FAMILY_10_SUPPORT

This option removes the code which implements support for family 10h processors. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FAMILY_12_SUPPORT

This option removes the code which implements support for family 12h processors. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FAMILY_14_SUPPORT

This option removes the code which implements support for family 14h processors. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FAMILY_15_SUPPORT

This option removes the code which implements support for family 15h processors, models 00h through 0Fh. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FAMILY_15_MODEL_1X_SUPPORT

This option removes the code which implements support for family 15h processors, models 10h through 1Fh. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FAMILY_15_MODEL_3X_SUPPORT

This option removes the code which implements support for family 15h processors, models 30h through 3Fh. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_AM3_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for AM3 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_ASB2_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for ASB2 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_C32_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for C32 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FM1_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FM1 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FP1_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FP1 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FS1_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FS1 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FT1_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FT1 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FP3_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FP3 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FM2_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FM2 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_FS2_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for FS2 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_G34_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for G34 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_S1G3_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for S1g3 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_S1G4_SOCKET_SUPPORT

This option removes the code which implements support for processors packaged for S1g4 sockets. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_CONNECTED_STANDBY

This option removes the code which implements support for Connected Standby. Some customers will prefer to use secure s3 instead of Connected Standby and will want to be able to completely remove the code. If the platform does not include support, then this option has no effect.

BLDOPT_REMOVE_S3_SUPPORT

This option removes the code which implements support for s3. Some customers prefer alternative power management like Connected Standby, or have ‘always on’ units.

16.2 Build Configuration Elements

Build configuration elements are settings that specify platform-specific boundaries chosen by the motherboard designer. These settings do not affect the size of code like the build options above. These elements are used to import into the build the data values that describe the individual motherboard on which the code build executes.

The user specifies build configuration elements in the <plat>Options.c source file local to their project. Use of these configuration elements is completely optional; the specified defaults are used if the element is not specified in the file.

16.2.1 Processor and General Elements

BLDCFG_PCI_MMIO_BASE

Specifies an address to be used as the extended MMIO PCI configuration base address. The address may be up to 48 bits and must be aligned on “BLDCFG_PCI_MMIO_SIZE” . The default is zero, which disables the extended MMIO method.

BLDCFG_PCI_MMIO_SIZE

The size of the Extended MMIO region. Must be 1, 2, 4, 8, 16, 32, 64, 128, or 256 MBytes. Default is zero.

BLDCFG_REMOVE_ACPI_PSTATES_PPC

Remove the ACPI PPC table from the ACPI table output.

BLDCFG_REMOVE_ACPI_PSTATES_PCT

Remove the ACPI PCT table from the ACPI table output.

BLDCFG_REMOVE_ACPI_PSTATES_PSD

Remove the ACPI PSD table from the ACPI table output.

BLDCFG_REMOVE_ACPI_PSTATES_PSS

Remove the ACPI PSS table from the ACPI table output.

BLDCFG_REMOVE_ACPI_PSTATES_XPSS

Remove the ACPI XPSS table from the ACPI table output.

BLDCFG_ACPI_SET_OEM_ID

Set the OEM ID field in ACPI table outputs to this string. The string must conform to the ACPI rules for the OEM ID field. The default value is “AMD”.

BLDCFG_ACPI_SET_OEM_TABLE_ID

Set the OEM TABLE ID field in ACPI table outputs to this string. The string must conform to the ACPI rules for the OEM TABLE ID field. By default arbitrary table names will be used.

BLDCFG_VRM_CURRENT_LIMIT

This value applies to the CoreVrm property set. This is the maximum current that the voltage regulators are capable of providing to the executing core’s socket on a continuous basis. This a numeric value and must be in milliamperes. For example, a current limit of 12.6 amperes is represented as:

```
#define BLDCFG_VRM_CURRENT_LIMIT 12600
```

BLDCFG_VRM_NB_CURRENT_LIMIT

This value applies to the NbVrm property set. This is the maximum current that the voltage regulators are capable of providing to the processor Northbridge on a continuous basis. This a numeric value and must be in milliamperes. For example, a current limit of 12.6 amperes is represented as:

```
#define BLDCFG_VRM_NB_CURRENT_LIMIT 12600
```

BLDCFG_VRM_LOW_POWER_THRESHOLD

This value applies to the CoreVrm property set. Some models of processor support an ability (output signal) to indicate when the processor is in a low-power state. This feature can be used by the power regulator to place itself into a more power-efficient mode. This parameter specifies the maximum current in mA that the power regulator can source in its power efficiency mode. This value is translated into a PState threshold. When the processor enters a PState under this threshold, the power regulator is signaled that it can enter its power efficiency mode. Note that some VRMs may require this value to reflect a current in-rush.

The default action is to set a value of zero, indicating to disable the feature. To specify a threshold, the value must be specified in **milli-amps**. For example, a current of 2.75 amps is represented as:

```
#define BLDCFG_VRM_LOW_POWER_THRESHOLD 2750
```

BLDCFG_VRM_NB_LOW_POWER_THRESHOLD

This value applies to the NbVrm property set. Some models of processor support an ability (output signal) to indicate when the processor is in a low-power state. This feature can be used by the power regulator to place itself into a more power-efficient mode. This parameter specifies the maximum current in mA that the power regulator can source in its power efficiency mode. This value is translated into a PState threshold. When the processor enters a PState under this threshold, the power regulator is signaled that it can enter its power efficiency mode. Note that some VRMs may require this value to reflect a current in-rush.

The default action is to set a value of zero, indicating to disable the feature. To specify a threshold, the value must be specified in **milli-amps**. For example, a current of 2.75 amps is represented as:

```
#define BLDCFG_VRM_NB_LOW_POWER_THRESHOLD 2750
```

BLDCFG_VRM_SLEW_RATE

This value applies to the CoreVrm property set. This signifies the rate at which the executing core voltage regulators are capable of transitioning from one voltage level to the next, including any delay. This number is defined as the number of milli-volts per micro-second. For example, a slew rate of 2.5 mV/µS is represented as:

```
#define BLDCFG_VRM_SLEW_RATE 2500
```

BLDCFG_VRM_NB_SLEW_RATE

This value applies to the NbVrm property set. This signifies the rate at which the executing core voltage regulators are capable of transitioning from one voltage level to the next, including any delay. This number is defined as the number of milli-volts per micro-second. For example, a slew rate of 2.5 mV/µS is represented as:

```
#define BLDCFG_VRM_NB_SLEW_RATE 2500
```

BLDCFG_VRM_HIGH_SPEED_ENABLE

This value applies to the CoreVrm property set. This indicates that the VRM is high frequency or not. Set this according to the VRM for the board. The default is FALSE.

TRUE — The VRM supports high frequency operation.

FALSE — The VRM does not support high frequency operation.

Example:

```
#define BLDCFG_VRM_LINK_FREQUENCY TRUE
```

BLDCFG_VRM_NB_HIGH_SPEED_ENABLE

This value applies to the NbVrm property set. This indicates that the VRM is high frequency or not. Set this according to the VRM for the board. The default is FALSE.

TRUE — The VRM supports high frequency operation.

FALSE — The VRM does not support high frequency operation.

Example:

```
#define BLDCFG_VRM_NB_LINK_FREQUENCY TRUE
```

BLDCFG_VRM_MAXIMUM_CURRENT_LIMIT

This value applies to the CoreVrm property set. This is the maximum current that the voltage regulators are capable of providing to the executing core's socket for short amounts of time. This a numeric value and must be in milliamperes. For example, a current limit of 12.6 amperes is represented as:

```
#define BLDCFG_VRM_MAXIMUM_CURRENT_LIMIT 12600
```

BLDCFG_VRM_NB_MAXIMUM_CURRENT_LIMIT

This value applies to the NbVrm property set. This is the maximum current that the voltage regulators are capable of providing to the executing core's socket for short amounts of time. This a numeric value and must be in milliamperes. For example, a current limit of 12.6 amperes is represented as:

```
#define BLDCFG_VRM_NB_MAXIMUM_CURRENT_LIMIT 12600
```

BLDCFG_VRM_SVI_OCP_LEVEL

This value applies to the CoreVrm property set. This is the current level at which the VRM will reduce current in order to protect electrical components from damage. This a numeric value and must be in milliamperes. For example, a current limit of 18.9 amperes is represented as:

```
#define BLDCFG_VRM_SVI_OCP_LEVEL 18900
```

BLDCFG_VRM_NB_SVI_OCP_LEVEL

This value applies to the NbVrm property set. This is the current level at which the VRM will reduce current in order to protect electrical components from damage. This a numeric value and must be in milliamperes. For example, a current limit of 18.9 amperes is represented as:

```
#define BLDCFG_VRM_SVI_NB_OCP_LEVEL 18900
```

BLDCFG_AMD_PSTATE_CAP_VALUE (deprecated)

This item is deprecated; use BLDCFG_AMD_TDP_LIMIT instead.

BLDCFG_CPU_FREQUENCY_LIMIT

This element specifies a maximum frequency at which the CPU core should operate.

Zero (0) - (default) No limit is applied. The core will run to their fused maximum frequency.

N (non-zero) - This value will be used to limit boost and SW PStates. This value is presented in MegaHertz. For example, a limit of 1600Mhz is set as:

```
#define BLDCFG_CPU_FREQUENCY_LIMIT 1600
```

BLDCFG_FORCE_INDEPENDENT_PSD_OBJECT

This element selects whether P-States should be forced to be independent, as reported by the ACPI _PSD object. This setting can improve performance for OS which support this feature. The default value is FALSE, which results in a processor specific setting. Processors which do not default to independent, default to dependent.

TRUE — Report independent P-State control.

FALSE — The PSD will report the default P-State independence or dependence.

BLDCFG_PROCESSOR_SCOPE_NAME0

This element allows a customized choice for the first letter in the ACPI processor scope name. The default is “C”.

BLDCFG_PROCESSOR_SCOPE_NAME1

This element allows a customized choice for the second letter in the ACPI processor scope name. The default is “0”.

BLDCFG_PROCESSOR_SCOPE_IN_SB

This element specifies whether ACPI _PSS objects are defined in the system bus or processor scope.

TRUE — The objects will be under the _SB scope.

FALSE — The objects will be under the _PR scope (default).

BLDCFG_PLATFORM_NUM_IO_APICS

This element provides the number of IO APICs that are used on the motherboard. This is used by the software to appropriately assign APIC IDs to the processors, leaving room for the motherboard devices. The default setting is 3.

For example:

```
#define BLDCFG_PLATFORM_NUM_IO_APICS 4
```

BLDCFG_PLATFORM_CSTATE_MODE

This element specifies the C State operational mode. This can be used with processors which support C States other than C1e.

CStateModeDisabled — The feature is not enabled.

CStateModeC6 — (default) Enable C6 C State.

BLDCFG_PLATFORM_CSTATE_OPDATA

This element specifies some pertinent data needed for the operation of the C State feature. For CStateModeC6, this item is reserved.

BLDCFG_PLATFORM_CSTATE_IO_BASE_ADDRESS

This element specifies some pertinent data needed for the operation of the C State feature. This item specifies a free block of 8 consecutive bytes of I/O ports that can be used to allow the CPU to enter C States. This item should always be specified regardless of the C State mode selected. The default value of zero disables I/O C State transitions.

BLDCFG_ CPU_CONECTED_STANDBY_MODE

This element specifies how to enable the Connected Standby feature. Note the Connected Standby feature is an optional code block. This option will only be effective when the code is present, see "BLDOPT_REMOVE_CONNECTED_STANDBY" on page 181.

ConnectedStandbyAuto - (default) processors supporting Connected Standby will have it enabled.
ConnectedStandbyDisabled - force ConnectedStandby to be disabled. This does not affect the presence of the support code block.

BLDCFG_PLATFORM_CPB_MODE

This item allows Core Performance Boost (CPB) to be forced to disabled, even if the hardware provides this feature. By default, CPB is enabled for processors that support it.

CpbModeAuto - (default) processors with CPB will have it enabled.
CpbModeDisabled - force CPB to be disabled.

BLDCFG_HYBRID_BOOST_ENABLE

This element specifies to enable hybrid boost mode for processors which support it.

TRUE - The Hybrid Boost feature is enabled (default).
FALSE - The Hybrid Boost feature is not enabled.

BLDCFG_CORE_LEVELING_MODE

A multi-socket system can be populated with processors having various number of CPU cores. Many operating systems require a homogeneous system with regard to processor feature set and some operating systems are limited in how many cores they can support. This parameter allows control of how the number of CPU cores is leveled to meet the various requirements.

CORE_LEVEL_LOWEST—(default) also known as: Automatic mode. All processors are leveled to have the same number of cores. The number of cores is reduced to match the processor with the minimum number of cores.

CORE_LEVEL_TWO — Force all processors to use only two cores. This may be required by some operating systems that do not recognize processors with more than two cores. If any processor has only one core, then all processors are leveled to one core.

CORE_LEVEL_POWER_OF_TWO — Power of 2 mode. All processors are leveled to have the same number of cores. The number of cores is reduced to match the processor with the minimum number of cores, with an additional reduction to the nearest power of 2 (1, 2, 4, or 8). This is required by some operating systems.

CORE_LEVEL_NONE — No leveling performed. All processors are left at their natural number of cores. A very select few operating systems may support this configuration, which allows a system with processors having a variety of numbers of cores.

CORE_LEVEL_COMPUTE_UNIT — Level cores to one core per compute unit, with additional reduction to level all processors to match the processor with the minimum number of cores. For processors which do not have compute units, this is the same as CORE_LEVEL_LOWEST.

CORE_LEVEL_COMPUTE_UNIT_TWO — Level cores to two cores per compute unit, with additional reduction to level all processors to match the processor with the minimum number of cores. For processors which do not have compute units, this is the same as CORE_LEVEL_LOWEST.

CORE_LEVEL_COMPUTE_UNIT_THREE — Level cores to three cores per compute unit, with additional reduction to level all processors to match the processor with the minimum number of cores. For processors which do not have compute units, this is the same as CORE_LEVEL_LOWEST.

For general purpose usage it is recommended to use one of the above leveling modes. However, for a few very specific applications the explicit leveling controls below are available. Note that some processors may not be capable of supporting all of these options. For example, MCM processors can not have an odd number of cores. For an odd CORE_LEVEL_N, MCM processors will be leveled as though CORE_LEVEL_N+1 was chosen.

Processors with compute units disable all cores in an entire compute unit at a time, or on an MCM processor, two compute units at a time. For example, on an SCM processor with two cores per compute unit, the effective explicit levels are CORE_LEVEL_ONE, CORE_LEVEL_TWO, CORE_LEVEL_FOUR, CORE_LEVEL_SIX, and CORE_LEVEL_EIGHT. The same example for an MCM processor with two cores per compute unit has effective explicit levels of CORE_LEVEL_TWO, CORE_LEVEL_FOUR, CORE_LEVEL_EIGHT, and CORE_LEVEL_TWELVE.

CORE_LEVEL_ONE — Explicit leveling. Set leveling to one core per processor. MCM processors will be leveled as though CORE_LEVEL_TWO was chosen.

CORE_LEVEL_THREE — Explicit leveling. Set leveling to three cores. If any processor has less than three cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_FOUR was chosen.

CORE_LEVEL_FOUR — Explicit leveling. Set leveling to four cores. If any processor has less than four cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen.

CORE_LEVEL_FIVE — Explicit leveling. Set leveling to five cores. If any processor has less than five cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_SIX was chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_SIX — Explicit leveling. Set leveling to six cores. If any processor has less than six cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_SEVEN — Explicit leveling. Set leveling to seven cores. If any processor has less than seven cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_EIGHT was chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_EIGHT — Explicit leveling. Set leveling to eight cores. If any processor has less than eight cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen.

CORE_LEVEL_NINE — Explicit leveling. Set leveling to nine cores. If any processor has less than nine cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_TEN was chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_TEN — Explicit leveling. Set leveling to ten cores. If any processor has less than ten cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_ELEVEN — Explicit leveling. Set leveling to eleven cores. If any processor has less than eleven cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_TWELVE was chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_TWELVE — Explicit leveling. Set leveling to twelve cores. If any processor has less than twelve cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen.

CORE_LEVEL_THIRTEEN — Explicit leveling. Set leveling to thirteen cores. If any processor has less than thirteen cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_FOURTEEN was chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_FOURTEEN — Explicit leveling. Set leveling to fourteen cores. If any processor has less than fourteen cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

CORE_LEVEL_FIFTEEN — Explicit leveling. Set leveling to fifteen cores. If any processor has less than fifteen cores, then all processors are leveled as though CORE_LEVEL_LOWEST were chosen. MCM processors will be leveled as though CORE_LEVEL_LOWEST was chosen. Additionally, processors with compute units will be leveled to contain complete compute units.

Example:

```
#define BLDCFG_CORE_LEVELING_MODE CORE_LEVEL_TWO
```

BLDCFG_AP_MTRR_SETTINGS_LIST

The ApMtrrSettingsList allows the customization of the APs' Fixed-Sized MTRR settings. The MTRR settings are applied after early initialization before IBV takes control of the APs.

The default value is NULL, indicating that no MTRR override setting is provided. To provide an override setting list, assign the build element to the

name of a user defined settings table. For details, refer to the internal documentation, “Internal Documentation” on page 25.

Only the MTRRs listed in the user defined table will be overwritten; the settings for the other fixed-sized MTRRs will be set to default values. The default Fixed-Sized MTRR settings are:

```
0x00000 - 0x9FFFF : WB Memory
0xA0000 - 0xDFFFF : UC I/0
0xE0000 - 0xFFFFF : UC Memory
```

For example,

```
#define BLDCFG_AP_MTRR_SETTINGS_LIST &MyApMtrrsList
```

BLDCFG_PLATFORM_CONTROL_FLOW_MODE

This value is used to select the optimum flow control method for the platform. Considerations include Display Refresh, Isochronous Flow and IOMMU. The available values are listed in the AGESA.h file in the enumeration declaration for PLATFORM_CONTROL_FLOW. The default is Nfcm, for Normal Control Flow Mode.

Example:

```
#define BLDCFG_PLATFORM_CONTROL_FLOW_MODE Iommu
```

BLDCFG_USE_32_BYTE_REFRESH

This is a performance optimization setting and indicates whether or not to enable use of smaller refresh packets. This is a BOOLEAN value:

FALSE — Do not enable the feature (default).
TRUE — Activate the feature.

Example:

```
#define BLDCFG_USE_32_BYTE_REFRESH TRUE
```

BLDCFG_DRAM_DOUBLE_REFRESH_RATE

This is a performance optimization setting and indicates whether or not to refresh the memories at twice the rate calculated or as indicated in the SPD information. This is a BOOLEAN value:

FALSE — Use the standard refresh rate (default). {Example: 7.8us}
TRUE — Double the refresh rate (cut period in half). {Example: 3.9us}

Example:

```
#define BLDCFG_DRAM_DOUBLE_REFRESH_RATE TRUE
```

BLDCFG_USE_VARIABLE_MCT_ISOC_PRIORITY

This indicates whether or not to enable the memory controller to use special handling for ISOC transfers. This is a BOOLEAN value:

FALSE —Do not enable the feature (default).
TRUE — Activate the feature.

Example:

```
#define BLDCFG_USE_VARIABLE_MCT_ISOC_PRIORITY TRUE
```

BLDCFG_PERFORMANCE_HARDWARE_PREFETCHER

This value provides for advanced performance tuning by controlling the hardware prefetcher setting. Use the recommended setting for best general performance, but this item can allow for optimizing specific workloads. The settings below are ordered from all enabled to all disabled and a specific disable setting implies all disable settings above it in the list as well. For example, disabling the L1 prefetcher implies that hardware prefetcher training on software prefetches is also disabled.

HARDWARE_PREFETCHER_AUTO - Use the recommended setting for the processor. In most cases, the recommended setting is enabled.

DISABLE_HW_PREFETCHER_TRAINING_ON_SOFTWARE_PREFECTCHES - Use the recommended setting for the hardware prefetcher, but disable training on software prefetches.

DISABLE_L1_PREFETCHER - Use the recommended settings for the hardware prefetcher, but disable L1 prefetching and above.

DISABLE_L2_STRIDE_PREFETCHER - Use the recommended settings for the hardware prefetcher, but disable the L2 stride prefetecher and above.

DISABLE_HARDWARE_PREFETCH - Disable hardware prefetching.

DISABLE_L1_PREFETCHER_AND_HW_PREFETCHER_TRAINING_ON_SOFTWARE_PREFECTCHES (**deprecated**) - This setting is deprecated; it has the same effect as DISABLE_L1_PREFETCHER.

BLDCFG_PERFORMANCE_SOFTWARE_PREFETCHES

This value provides for advanced performance tuning by controlling the software prefetch instructions. Use the recommended setting for best general performance, but this item can allow for optimizing specific workloads.

SOFTWARE_PREFETCHES_AUTO - Use the recommended setting for the processor. In most cases, the recommended setting is enabled.

DISABLE_SOFTWARE_PREFETCHES - Disable software prefetches (convert software prefetch instructions to NOP).

BLDCFG_PERFORMANCE_DRAM_PREFETCHER

This value provides for advanced performance tuning by controlling the DRAM prefetcher setting. Use the recommended setting for best general performance, but this item can allow for optimizing specific workloads.

DRAM_PREFETCHER_AUTO - Use the recommended setting for the processor. In most cases, the recommended setting is enabled.
DISABLE_DRAM_PREFETCH_FOR_IO -Disable DRAM prefetching for I/O requests only.
DISABLE_DRAM_PREFETCH_FOR_CPU - Disable DRAM prefetching for requests from processor cores only.
DISABLE_DRAM_PREFETCHER - Disable DRAM prefetching.

BLDCFG_NB_PSTATES_SUPPORTED

This value indicates whether to enable processor northbridge (NB) P-States (as opposed to processor core P-States).

TRUE - Enable NB P-States, if supported by the processor (default).
FALSE - Disable NB P-States.

16.2.1.1 Thermal and Power Control Elements

BLDCFG_AMD_TDP_LIMIT

Specifies a maximum power limit for the platform. For processors with TDP limiting support the limit will use the TDP limiting if the configuration is supported. Otherwise, P-State capping will be used to implement the TDP limit. The default is zero, indicating not to limit TDP but to use the recommended settings. Otherwise, this is the integer number in milliwatts to be used as the TDP limit.

BLDCFG_DOCKED_TDP_HEADROOM

Specifies if there is additional heat dissipation available when the system is docked.

TRUE — (default) The motherboard complies with the AMD recommendations for mobile units. There is extra heat dissipation capability in the dock.

FALSE — There is no extra heat dissipation available.

Note: This feature applies to the APUs using the TurboDock feature (Kabini, Temash). Future APUs may deprecate this control to support new features, see the “Elements for DcTDP_V2.0” on page 194.

BLDCFG_BATTERY_BOOST_EN

Specifies if the Boost feature should be allowed while operating on battery power.

TRUE — (default) processor Boost states will be used.

FALSE — Save power by disabling Boost states while on battery power.

BLDCFG_BATTERY_BOOST_TUNE

Specifies the behavior of the Battery Based Boost algorithm. This parameter is valid only when BLDCFG_BATTERY_BOOST_ENABLE is set to True.

This parameter takes a numeric value between 0-100 to adjust the boost behavior allowed when the system is on battery power. A value of 100 (default) indicates to use the maximum boost allowed while in battery mode. A value less than 100 will reduce the amount of boosting used and may be used to trade-off boosting for battery life.

BLDCFG_NUM_GFX_CORES_ENABLED

Used to specify how many graphic compute units are to be enabled on the APU. This control allows designers to reduce the number of active compute units for thermal/power reasons.

This is a number between 0 and <N> (the maximum available conpute units). The default value of (-1) indicates no limit, use all available compute units.

BLDCFG_BAPM_ENABLE

Specifies the active state of the BAPM thermal control feature

TRUE — (default) the BAPM controls will be used.

FALSE — Disable the BAPM features.

16.2.1.1.1 Elements for DcTDP_V2.0

The algorithms used for power and thermal management continues to evolve. The Mullins APU expands the dynamic TDP controls and adds a new capability called Skin Temperature Adaptive Power Management (STAPM). Older APUs that share the same socket may also need to use the “BLDCFG_DOCKED_TDP_HEADROOM” control to support their features.

The DcTDP_V2.0 feature defines a 'four box' set of configurable control values. The current management parameters being enforced are selected by the indices of the box.

	AC	DC
Docked	TDP = 9 watts STAPM enabled = False STAPM power = (n/a) TSP = (n/a)	TDP = 9 watts STAPM enabled = False STAPM power = (n/a) TSP = (n/a)
UnDocked	TDP = (n/a) STAPM enabled = True STAPM power = 9 watts TSP = 5 watts	TDP = (n/a) STAPM enabled = True STAPM power = 0 (use pre-sets)) TSP = 0 (use pre-sets)
* The numbers used in this example are illustrative of a predicted configuration, but each OEM should determine the values appropriate for their platform. Please refer to *Thermal Design for Fanless Tablets and Infrastructure Road Map (IRM) Specification.*		

For more information about the thermal control features and algorithms, please refer to the Infrastructure Road Map (IRM) spec for the specific APU.

BLDCFG_TDP_AC_DOCKED

BLDCFG_TDP_DC_DOCKED

BLDCFG_TDP_AC_UNDOCKED

BLDCFG_TDP_DC_UNDOCKED

Specifies the TDP the systems should target when the unit is operating in the indicated box.

This is a numeric value representing the absolute TDP, in milliwatts. For example, a target TDP of 4.5 watts is represented as:

```
#define BLDCFG_TDP_DC_UNDOCKED 4500
```

The default value is zero (0), meaning the unit will use the pre-set values for the part.

BLDCFG_TSP_AC_DOCKED

BLDCFG_TSP_DC_DOCKED

BLDCFG_TSP_AC_UNDOCKED

BLDCFG_TSP_DC_UNDOCKED

Specifies the Thermally Significant Power (TSP) the systems should target when the unit is operating in the indicated box. The TSP is the recommended target power level for designing skin temperature limited fanless platforms

This parameter is used only when STAPM is enabled. For more information on STAPM and TSP refer to the *Thermal Design for Fanless Tablets*.This is a numeric value representing the absolute TSP, in milliwatts. For example, a target TSP of 4.5 watts is represented as:

```
#define BLDCFG_TSP_DC_UNDOCKED 4500
```

The default value is zero (0), meaning the unit will use the pre-set values for the part. For information about the default settings of this parameter, refer to the *Infrastructure Road Map (IRM) Specification*.

BLDCFG_STAPM_EN_AC_DOCKED

BLDCFG_STAPM_EN_DC_DOCKED

BLDCFG_STAPM_EN_AC_UNDOCKED

BLDCFG_STAPM_EN_DC_UNDOCKED

Select whether or not the STAPM process should be active when the unit is operating in the indicated box. When the STAPM feature is active, the unit will follow the TSP for guidance on actions. When the STAPM feature is disabled, the unit will follow the TDP for guidance on actions.

This is a boolean value.

TRUE - Enable the STAPM process in this box (default).
FALSE - Disable STAPM for this box.

BLDCFG_STAPM_POWER_AC_DOCKED

BLDCFG_STAPM_POWER_DC_DOCKED

BLDCFG_STAPM_POWER_AC_UNDOCKED

BLDCFG_STAPM_POWER_DC_UNDOCKED

This parameter specifies the maximum power that the APU is allowed to consume as long as the skin temperature of the system stays within the specified limits. This parameter is used only when STAPM is enabled. For STAPM-supported systems, the thermal solution should be designed to STAPM Power levels at maximum die temperature (Tdie max). For more information on this parameter, refer to the *Thermal Design for Fanless Tablets*.

This is a numeric value representing the absolute power, in milliwatts. For example, a target power of 9 watts is represented as:

```
#define BLDCFG_STAPM_POWER_DC_UNDOCKED 9000
```

A value of 0 (default) means that pre-set values for the part will be used. For information about the pre-set settings of the parts, refer to the *Infrastructure Road Map (IRM) Specification*.

16.2.2 Graphics and PCIe® Elements

BLDCFG_CFG_GNB_HD_AUDIO

This indicates whether to enable HD Audio.

FALSE - Disabled (default).
TRUE - Enable HD audio.

BLDCFG_MAX_NUM_AUDIO_ENDPOINTS

This option defines the mamimum number of audio endpoints supported by the platform. To be applied, this value must be less than the number of display pipes supported by the APU. This value should be defined if the platform supports a number of audio endpoints that is smaller than the number supported by the APU.

This is a numeric value:
N - The number of active display pipes, or audio end points on the host platform.
255 - Max value (Default). Using the default, or not specifying a value will use all endpoints supported on the APU.

BLDCFG_GNB_AZ_I2SBUS_SELECT

From the Carrizo APU and beyond, the pins used to connect the (former) Azalia bus can be configured to use the new I2SBus protocol. This option specifies the pin functions.

GnbAcpAzalia - (default) Use the standard Azalia protocol.
GnbAcpI2sBus - Use the industry standard I^2S Bus protocol.

BLDCFG_GNB_AZ_I2SBUS_PIN_CONFIG

From the Carrizo APU and beyond, the pins used to connect the (former) Azalia bus can be configured to use the new I^2S Bus protocol. When *BLDCFG_GNB_AZ_I2SBUS_SELECT* is set select the I^2S Bus protocol, this additionaloption specifies the I^2S protocol.

GnbAcp4Tx4RxBluetooth - (default) .
GnbAcp2Tx4RxBluetooth - .
GnbAcp6Tx4RxBluetooth - .

BLDCFG_CFG_ABM_SUPPORT

This indicates whether to enable ABM support for display panel back light.

FALSE - Disabled (default).
TRUE - Enable ABM.

BLDCFG_CFG_DYNAMIC_REFRESH_RATE

This provides the dynamic display refresh rate. This value is dependent on the display panel used.

If an embedded panel supports a contiguous range of refresh rates but the minimum is not described in its EDID, then this parameter allows the platform to expose the capability. The value here defines the minimum refresh rate, while the maximum refresh rate is obtained from the EDID.

This value is dependent on the display panel used .

This value has a different meaning for the newer APUs. For Kabini and older APU products, this definition is being deprecated:

0 - Either the embedded panel carries the information in its EDID, or the Platform doesn't need this feature .
Non-zero: choose one of the following bits to indicate the minimum refresh rate that the panel supports:

SUPPORTED_LCD_REFRESHRATE_30Hz	0x0004
SUPPORTED_LCD_REFRESHRATE_40Hz	0x0008
SUPPORTED_LCD_REFRESHRATE_50Hz	0x0010
SUPPORTED_LCD_REFRESHRATE_60Hz	0x0020
SUPPORTED_LCD_REFRESHRATE_48Hz	0x0040

For Kaveri and onward APU products:
0 - Either the embedded panel carries the information in its EDID, or the Platform doesn't need this feature .
N - A numeric value to indicate the minimum refresh rate that the panel supports.

BLDCFG_CFG_LCD_BACK_LIGHT_CONTROL

This provides support for PWM back light control. The value is dependent on the PWM controller and platform characterization.

0 - No display rate specified.
N - Use this number, N, for the PWM control.

BLDCFG_ACP_SIZE

This element specifies the size of the Audio Co-Processor (ACP) region. ACP is an audio signal processor used to accelerate audio encode and decode.

NO_ACP_SIZE	Do not use ACP (default).
ACP_SIZE_2MB	ACP is allocated 2MBytes region.
ACP_SIZE_4MB	ACP is allocated 4 MBytes region.

BLDCFG_STEREO_3D_PINOUT

Specifies the pin index associated with the Stereo 3D feature.

0 - Stereo 3D is disabled (default).
1 - Use processor pin HPD1.
2 - Use processor pin HPD2.
3 - Use processor pin HPD3.
4 - Use processor pin HPD4.
5 - Use processor pin HPD5.
6 - Use processor pin HPD6.

BLDCFG_IGPU_SUBSYSTEM_ID

This item specifies a customized PCIe® Subsystem ID for the processor's Internal Graphics Processing Unit (IGPU) device. The default is zero.

BLDCFG_IGPU_HD_AUDIO_SUBSYSTEM_ID

This item specifies a customized PCIe® Subsystem ID for the IGPU's HD Audio device. The default is zero.

BLDCFG_APU_PCIE_PORTS_SUBSYSTEM_ID

This item specifies a customized PCIe® Subsystem ID for the processor's PCIe® ports. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_PCIE_TRAINING_ALGORITHM

This item selects the training algorithm for PCIe® links. Customizing the algorithm used may provide better boot performance, depending on system design. Refer to the internal documentation for the latest and most detailed information ("Internal Documentation" on page 25).

PcieTrainingStandard - Standard algorithm (default).
PcieTrainingDistributed - Training is distributed over multiple entry points. This potentially improves S3 resume time, but PCIe® devices are not accessible until later in the boot sequence.

BLDCFG_IOMMU_SUPPORT

This indicates whether to enable IOMMU support.

FALSE - Disable IOMMU support.
TRUE - Enable IOMMU support for processors with IOMMU (default).

BLDCFG_IOMMU_EXCLUSION_RANGE_LIST

This specifies the IOMMU exclusion ranges for I/O Virtualization.

Details on how to create the list content can be found in "IOMMU Exclusion Range Descriptor" on page 331.

The default value is NULL, indicating that no list is provided. To provide a user-defined list, assign the above name to the name of the user-created data table.

For example:

```
#define BLDCFG_IOMMU_EXCLUSION_RANGE_LIST MyIommuExclusionList
```

BLDCFG_GNB_IOAPIC_ADDRESS

This specifies the IOAPIC base address for the processor IOAPIC. The value is a 64 bit address aligned to 256 byte boundaries. The default value of NULL signifies that the host environment will set the base address itself.

BLDCFG_GFX_LVDS_SPREAD_SPECTRUM

This specifies the LVDS spread spectrum value.

0 - Spread spectrum is disabled (default).
n - Values greater than zero specify a setting of 0.0n %. For example, 3 corresponds to a setting of 0.03%.

BLDCFG_GFX_LVDS_SPREAD_SPECTRUM_RATE

This specifies the spread spectrum frequency, if "BLDCFG_GFX_LVDS_SPREAD_SPECTRUM" is not disabled.

0 - 40 kHz (default).
n - Values greater than zero specify a setting of (40 + (n * 10)) kHz. For example, 2 corresponds to 60 kHz.

BLDCFG_LVDS_POWER_ON_SEQ_DIGON_TO_DE

This specifies the LVDS power up sequence time for the delay from DIGON active to Data Enable (DE) active.

0 - Use the video BIOS default (default). The video BIOS default is 8 ms, or 32 ms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_POWER_ON_SEQ_DE_TO_VARY_BL

This specifies the LVDS power up sequence time for the delay from Data Enable (DE) active to Vary Back Light (Vary BL) active.

0 - Use the video BIOS default (default). The video BIOS default is 90 ms, or 360 ms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_POWER_ON_SEQ_DE_TO_DIGON

This specifies the LVDS power up sequence time for the delay from active to active.

0 - Use the video BIOS default (default). The video BIOS default is 8 ms, or 32 ms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_POWERS_ON_SEQ_VARY_BL_TO_DE

This specifies the LVDS power up sequence time for the delay from active to active.

0 - Use the video BIOS default (default). The video BIOS default is 8 ms, or 32 ms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_POWER_ON_SEQ_ON_TO_OFF_DELAY

This specifies the LVDS power up sequence time for the delay from active to active.

0 - Use the video BIOS default (default). The video BIOS default is 8 ms, or 32 ms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_POWER_ON_SEQ_VARY_BL_TO_BLON

This specifies the LVDS power up sequence time for the delay from active to active.

0 - Use the video BIOS default (default). The video BIOS default is 8 ms, or 32 msms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_POWER_ON_SEQ_BLON_TO_VARY_BL

This specifies the LVDS power up sequence time for the delay from active to active.

0 - Use the video BIOS default (default). The video BIOS default is 8 ms, or 32 ms.
n - Values other than zero specify a setting of (4 * n) milliseconds time delay.

BLDCFG_LVDS_MAX_PIXEL_CLOCK_FREQ

This specifies the maximum pixel clock frequency.

0 - use the video BIOS default (default). The video BIOS default is 85 MHz.
n - Values other than zero specify a setting of n MHz.

BLDCFG_LCD_BIT_DEPTH_CONTROL_VALUE

This specifies the bit depth control settings, which can be used to tune display panel performance.

0 - Use video BIOS default (default). The video BIOS will auto detect the best settings to use.

n - Values other than zero specify the low level control settings.

BLDCFG_LVDS_24BBP_PANEL_MODE

This specifies the LVDS 24 BBP mode.

0 - Use LDI mode (default).

1 - Use FPDI mode.

BLDCFG_LVDS_MISC_888_FPDI_MODE

This specifies a miscellaneous control for LVDS.

TRUE - Use FPDI mode.
FALSE - Use LDI mode (default).

BLDCFG_LVDS_MISC_DL_CH_SWAP

This specifies a miscellaneous control for LVDS.

TRUE - The upper and lower channel are swapped.
FALSE - The upper and lower channel are not swapped (default).

BLDCFG_LVDS_MISC_VSYNC_ACTIVE_LOW

This specifies a miscellaneous control for LVDS.

TRUE - The VSYNC signal is active low.
FALSE - The VSYNC signal is active high (default).

BLDCFG_LVDS_MISC_HSYNC_ACTIVE_LOW

This specifies a miscellaneous control for LVDS.

TRUE - The HSYNC signal is active low.
FALSE - The HSYNC signal is active high (default).

BLDCFG_LVDS_MISC_BLON_ACTIVE_LOW

This specifies a miscellaneous control for LVDS.

TRUE - The BLON signal is active low.
FALSE - The BLON signal is active high (default).

BLDCFG_LVDS_MISC_VOLT_OVERWRITE_ENABLE

This specifies the output voltage adjustment, which may be needed with Travis.

FALSE - Use recommended setting (default).
TRUE - Overwrite the output voltage using LVDS Volt Adjustment.

BLDCFG_LVDS_MISC_VOLT_ADJUSTMENT

This specifies the output voltage adjustment to be made if enabled, see “BLDCFG_LVDS_MISC_VOLT_OVERWRITE_ENABLE” on page 202.

BLDCFG_GNB_THERMAL_SENSOR_CORRECTION

This specifies an increment to the default temperature junction (Tj) Offset. This control is for calibrating the on-chip thermal sensor. The thermal sensor correction value corrects for natural sensor variances inherent in the manufacturing process. See the processor's Thermal Design Guide for details.

BLDCFG_HTC_TEMPERATURE_LIMIT

This control specifies the temperature at which the processor's hardware thermal control (HTC) feature is activated. The HTC feature reduces the power consumption and performance of the processor when this temperature limit is reached to prevent further heating. The HTC feature has a Global ('whole chip') component and also a Local ('cores only') component.

This control (HTC_G Limit) is the 'Global' component which includes all of the sub-systems present on the processor.

If set, (non-zero) then the Local HTC temperature value must also be set, see "BLDCFG_LHTC_TEMPERATURE_LIMIT" on page 203. The temperature limit is specified in tenth of degrees centigrade increments (0.1). Values below 520 (52.0^{o}C) are ignored. Values above the factory default setting for the processor are ignored. See the processor's Thermal Design Guide for details.

0 - Use the factory setting. (default).
1.. 519 - invalid. Values in this range will be ignored. The factory setting will be used.
520..HTC_G - (52.0 to (HTC_G/10)^{o}C) specifies the temperature at which the HTC feature is to be invoked.

BLDCFG_LHTC_TEMPERATURE_LIMIT

This control applies only when the HTC_G limit is set manually (is non-zero). See "BLDCFG_HTC_TEMPERATURE_LIMIT" on page 203.

This control specifies the temperature at which the processor's hardware thermal control (HTC) feature is activated. The HTC feature reduces the power consumption and performance of the processor when this temperature limit is reached to prevent further heating. The HTC feature has a Global ('whole chip') component and also a Local ('cores only') component.

This control (HTC_L Limit) is the 'Local' component which includes only the compute unit cores present on the processor. This control allows a more fine-tuned handling of the thermal state.

This value is specified in tenth of degrees centigrade increments (0.1) and is recommended to be 3 to 5^{o}C below the value for HTC_G limit. Values above the HTC_L factory default setting for the processor are ignored. See the processor's Thermal Design Guide for details.

0 - Use the factory setting. (default).

1.. 519 - invalid. values in this range will be ignored. The default setting will be used.
520..HTC_L - (52.0 to (HTC_L/10)^oC) specifies the temperature at which the HTC feature is to be invoked.

BLDCFG_IGPU_ENABLE_DISABLE_POLICY

This specifies policies for enable or disable of the processor's internal graphics processing unit (GPU) when discrete graphics cards are present.

0 - Disable the internal graphics if the graphics card is not an AMD PCIe® graphics card (default).
1 - Disable the internal graphics if any graphics card is present.

BLDCFG_PCIE_REFCLK_SPREAD_SPECTRUM

When the platform has spread spectrum enabled on the PCIe® reference clock, this item should be used to provide that setting.

0 - Spread spectrum is not enabled for the platform PCIe® reference clock (default).
n - Spread spectrum is enabled at 0.0n %.

BLDCFG_REMOTE_DISPLAY_SUPPORT

This specifies whether remote, wireless display is supported.

FALSE - Do not support wireless display (default).
TRUE - Support wireless display.

BLDCFG_GPU_FREQUENCY_LIMIT

This specifies the maximum GPU frequency to be allowed, in MHz. This value is used to disable GPU power states that would exceed this frequency. This control may be used to improve (reduce) power consumption.

0 - (default) No limit is applied.
n - The GPU power states will be restricted to this limit. Power states using a frequency grater than this value will be disbled.

BLDCFG_DISPLAY_MISC_VBIOS_FAST_BOOT_ENABLE

This specifies whether the video BIOS software should skip display detection steps in order to reduce boot time.

FALSE - Fast boot is disabled (default).
TRUE - Fast boot is enabled.

BLDCFG_USE_SYNCFLOOD_AS_NMI

This specifies the function of the SYNCFLOODNMI# pin.

FALSE - The pin will be defined for sync flood (default).

TRUE - The pin will be defined for NMI.

16.2.3 Memory Elements

BLDCFG_MEMORY_BUS_FREQUENCY_LIMIT

This is the maximum memory clock at which the platform memory busses are capable of performing.

DDR400_FREQUENCY	DDR533_FREQUENCY
DDR667_FREQUENCY	DDR800_FREQUENCY (default)
DDR1066_FREQUENCY	DDR1333_FREQUENCY
DDR1600_FREQUENCY	DDR1866_FREQUENCY
DDR2100_FREQUENCY	DDR2133_FREQUENCY
DDR2400_FREQUENCY	

Example:

```
#define BLDCFG_MEMORY_BUS_FREQUENCY_LIMIT DDR1066_FREQUENCY
```

BLDCFG_MEMORY_MODE_UNGANGED

The platform should be set to use the memory channel unganged mode.

TRUE (default)	FALSE

Example:

```
#define BLDCFG_MEMORY_MODE_UNGANGED FALSE
```

BLDCFG_MEMORY_QUAD_RANK_CAPABLE

This is a platform-specific setting indicating that the memory slots are capable of supporting Quad Rank DIMMs.

TRUE (default)	FALSE

Example:

```
#define BLDCFG_MEMORY_QUAD_RANK_CAPABLE FALSE
```

BLDCFG_MEMORY_QUADRANK_TYPE

If the platform sets QRankCapable, then the Quad Rank DIMM slot type is one of the following:

`QUADRANK_REGISTERED`	(4-Rank Registered DIMMs) (default)
`QUADRANK_UNBUFFERED`	(Unbuffered SO-DIMMs)

If the platform is not indicated to be Quad Rank capable, then this element has no effect.

Example:

```
#define BLDCFG_MEMORY_QUADRANK_TYPE QUADRANK_REGISTERED
```

BLDCFG_MEMORY_SODIMM_CAPABLE

Specifies if the platform is designed to be capable of supporting SoDIMMs.

TRUE FALSE (default)

Example:

```
#define BLDCFG_MEMORY_SODIMM_CAPABLE TRUE
```

BLDCFG_MEMORY_LRDIMM_CAPABLE

Specifies if the platform is designed to be capable of supporting LRDIMMs. This item has no effect when "BLDOPT_REMOVE_LRDIMMS_SUPPORT" is TRUE.

TRUE FALSE (default)

Example:

```
#define BLDCFG_MEMORY_SODIMM_CAPABLE TRUE
```

BLDCFG_MEMORY_ENABLE_BANK_INTERLEAVING

Specifies if the system should use DRAM bank (also known as chip-select) interleaving. If the build option is set to include the code for this feature, then this setting activates the feature. If the feature code is removed from the build, then this element has no effect.
The AMD-recommended setting is indicated by the default setting. Changes to this setting must be based on individual platform testing.

TRUE =enable FALSE =disable (default)

Example:

```
#define BLDCFG_MEMORY_ENABLE_BANK_INTERLEAVING TRUE
```

BLDCFG_MEMORY_ENABLE_NODE_INTERLEAVING

Specifies if the system should use memory node interleaving. If the build option is set to include the code for this feature, then this setting activates the feature. If the feature code is removed from the build, then this element has no effect.
The AMD-recommended setting is indicated by the default setting. Changes to this setting must be based on individual platform testing.

TRUE = enable FALSE = disable (default)

Example:

```
#define BLDCFG_MEMORY_ENABLE_NODE_INTERLEAVING TRUE
```

BLDCFG_MEMORY_CHANNEL_INTERLEAVING

Specifies if the system should use memory channel interleaving for performance tuning. If the build option is set to include the code for this feature, then this setting activates the feature. If the feature code is removed from the build, then this element has no effect. Also, this option has no effect if the channel ganged mode is selected.
The AMD-recommended setting is indicated by the default setting. Changes to this setting must be based on individual platform testing.

TRUE = enable (default) FALSE = disable

Example:

```
#define BLDCFG_MEMORY_ENABLE_CHANNEL_INTERLEAVING TRUE
```

BLDCFG_MEMORY_POWER_DOWN

This feature conserves power in the DIMMs by placing them into self-refresh when the memory on a channel is not actively being accessed. They quickly exit self-refresh upon the next processor access.

TRUE =enable FALSE =disable (default)

Example:

```
#define BLDCFG_MEMORY_POWER_DOWN TRUE
```

BLDCFG_POWER_DOWN_MODE

This is a power-usage control that performs memory clock enable-based power-down control. If the platform enables power-down capability, then this element describes the platform method chosen, which is one of the following:

```
POWER_DOWN_BY_CHANNEL
POWER_DOWN_BY_CHIP_SELECT
POWER_DOWN_MODE_AUTO (default)
```

If the platform is not indicated to have power-down capability, then this element has no affect.

Example:

```
#define BLDCFG_POWER_DOWN_MODE POWER_DOWN_BY_CHIP_SELECT
```

BLDCFG_ONLINE_SPARE

This feature is recommended only for expert users and is described in the *BIOS and Kernel Developer's Guides (BKDG)*.

TRUE = enable Spare FALSE = disable Spare (default)

Example:

```
#define BLDCFG_ONLINE_SPARE TRUE
```

BLDCFG_MEMORY_PARITY_ENABLE

Parity is an error-detection tool available on some DIMMs. This control specifies whether or not the memory controller should activate parity checking. The feature is invoked only if the DIMMs present are all capable of supporting the parity detection.

TRUE = enable FALSE = disable (default)

Example:

```
#define BLDCFG_MEMORY_PARITY_ENABLE TRUE
```

BLDCFG_BANK_SWIZZLE

Address swizzle is a performance fine-tuning element that swaps some address lines. See the BKDG for description.

TRUE - enable (default) FALSE - disable

Example:

```
#define BLDCFG_BANK_SWIZZLE FALSE
```

BLDCFG_TIMING_MODE_SELECT

Allows the platform to select how the determination is made for the memory clock that is to be used in the system. (See also BLDCFG_MEMORY_CLOCK_SELECT.)

TIMING_MODE_AUTO—The AGESA™ software calculates the best memory clock rate (default).

TIMING_MODE_LIMITED—The AGESA™ software calculates the best memory clock, restricted by a maximum user limit provided in BLDCFG_MEMORY_CLOCK.

TIMING_MODE_SPECIFIC—The platform specifies the clock rate, which is provided in BLDCFG_MEMORY_CLOCK.

Example:

```
#define BLDCFG_TIMING_MODE_SELECT TIMING_MODE_LIMITED
```

BLDCFG_MEMORY_CLOCK_SELECT

This is the specified memory clock to be used in the system as determined by the selected mode. (See also BLDCFG_TIMING_MODE_SELECT.)

DDR400_FREQUENCY DDR533_FREQUENCY
DDR667_FREQUENCY DDR800_FREQUENCY

DDR1066_FREQUENCY DDR1333_FREQUENCY
DDR1600_FREQUENCY DDR1866_FREQUENCY
DDR2100_FREQUENCY DDR2133_FREQUENCY
DDR2400_FREQUENCY
The default value used depends upon the type of memory present.
DDR2 memories use a default value of DDR400_FREQUENCY.
DDR3 memories use a default value of DDR800_FREQUENCY.

BLDCFG_DQS_TRAINING_CONTROL

DQS signal timing training control is a platform selection to specify whether the automatic timing training routine is desired or a set of pre-defined timing values should be used. This can provide boot speed improvement by bypassing the active training algorithm and using previously stored values.

TRUE = perform active DQS training (default)
FALSE = skip DQS training

This feature is recommended only for expert users. Further details can be found in “Customizing the Environment—Library Functions” on page 219.

Example:

```
#define BLDCFG_DQS_TRAINING_CONTROL FALSE
```

BLDCFG_IGNORE_SPD_CHECKSUM

When the checksum of the SPD record fails, the typical action is to drop the DIMM and not attempt to configure it into the system. The software indicates via return code that this condition has occurred.

Under certain conditions, the platform can decide to accept the associated risk and choose to ignore the SPD checksum.

TRUE — Ignore faulty SPD checksum and continue DIMM initialization.
FALSE — If the checksum of an SPD is found to be invalid, then the memory initialization process is terminated for that DIMM (default).

Example:

```
#define BLDCFG_IGNORE_SPD_CHECKSUM TRUE
```

BLDCFG_USE_BURST_MODE

This is a performance fine-tuning element. The effect is described in the BKDG. This element may not be available in all families.

TRUE — enable (4-beat burst when width is 64 bits)
FALSE — disable (default)

Example:

```
#define BLDCFG_USE_BURST_MODE TRUE
```

BLDCFG_MEMORY_ALL_CLOCKS_ON

This is a power-usage control. To save power, unused memory clocks are disabled. Some platforms may not prefer to use this feature.

TRUE = Enable all memclocks whether they are used or not.
FALSE = Unused memclocks are disabled (default).

Example:

```
#define BLDCFG_MEMORY_ALL_CLOCKS_ON TRUE
```

BLDCFG_DDR_PHY_DLL_BYPASS_MODE

This element enables a low power DDR bus operation mode, which is useful for soldered down memory subsystems that follow low power design guidelines. This feature will limit the maximum DDR rate and should be disabled for high performance memory subsystem designs. This feature should be disabled for systems with DIMM slots rather than soldered down memory.

TRUE - Enable (default).
FALSE - Disable.

BLDCFG_MEMORY_PHY_VOLTAGE

This element specifies the voltage used to supply the Memory PHY circuits. This is a platform specific part of the design available for controling power consumption. For more information, please refer to the processor power specification. Note that the platform may be requested to change the PHY voltage via the "AgesaHookBeforeDramInit" using the parameter "VddpVddrVoltage" on page 116.

VOLT0_95 - The platform design specifies use of the lower voltage to save power.
VOLT1_05 - (default) Use the standard required voltage.

BLDCFG_ENABLE_ECC_FEATURE

This turns on the correction action and enables the ability of the MCA sub-system to report errors. It does not activate the MCA error report interrupts. If the build option is set to include the code for the ECC feature, then this setting activates the feature. If the feature code is removed from the build, then this element has no affect.

TRUE = enable (default)
FALSE = disable

Example:

```
#define BLDCFG_ENABLE_ECC_FEATURE FALSE
```

BLDCFG_ECC_REDIRECTION

DRAM ECC redirection is a data-protection feature. Redirection is a special ECC feature that enables the scrubber to immediately scrub any address in which a correctable error is discovered.
The AMD recommended setting is indicated by the default setting. Changes to this setting must be based on individual platform testing.

TRUE = enable FALSE = disable (default)

Example:

```
#define BLDCFG_ECC_REDIRECTION TRUE
```

BLDCFG_SCRUB_DRAM_RATE

This value selects how often the ECC background scrubber makes a pass through the DRAM. This is a numeric value and the value can vary from processor family to family.
0 = Disabled (default) x_x_x = Scrub Rate, see the BKDG.

Example:

```
#define BLDCFG_SCRUB_DRAM_RATE 0x0f
```

BLDCFG_SCRUB_L2_RATE

This value selects how often the ECC background scrubber makes a pass through the L2 cache. This is a numeric value and the value can vary from processor family to family.
0 =Disabled (default) x_x_x =Scrub Rate, see the BKDG.

Example:

```
#define BLDCFG_SCRUB_DRAM_RATE 0x0f
```

BLDCFG_SCRUB_L3_RATE

This value selects how often the ECC background scrubber makes a pass through the L3 cache. This is a numeric value and the value can vary from processor family to family and may not be available on all families.
0 =Disabled (default) x_x_x =Scrub Rate, see the BKDG.

Example:

```
#define BLDCFG_SCRUB_DRAM_RATE 0x0f
```

BLDCFG_SCRUB_IC_RATE

This value selects how often the ECC background scrubber makes a pass through the IC (Instruction Cache). This is a numeric value and the value can vary from processor family to family.
0 = Disabled (default) x_x_x =Scrub Rate, see the BKDG.

Example:

```
#define BLDCFG_SCRUB_DRAM_RATE 0x0f
```

BLDCFG_SCRUB_DC_RATE

This value selects how often the ECC background scrubber makes a pass through the DC (Data Cache). This is a numeric value and the value can vary from processor family to family.
0 = Disabled (default) x_x_x = Scrub Rate, see the BKDG.

Example:

```
#define BLDCFG_SCRUB_DRAM_RATE 0x0f
```

BLDCFG_ECC_SYNC_FLOOD

This indicates whether or not to cause a sync flood in the system when uncorrectable ECC errors are detected. This is a BOOLEAN value:

FALSE = Do not enable the MCA feature (default).
TRUE = Please see the BKDG for a description of the use of the Sync Flood and why a platform may choose to make use of this feature.

Example:

```
#define BLDCFG_ECC_SYNC_FLOOD TRUE
```

BLDCFG_ECC_SYMBOL_SIZE

This is an error-detection control. Please see the BKDG. for a description. This is a numeric value of 0, 4, or 8. Zero means to use the BKDG recommendation. The default is 4.

Example:

```
#define BLCFG_ECC_SYMBOL_SIZE 0x00
```

BLDCFG_UMA_ALLOCATION_MODE

Supply the uma memory allocation mode build time customization, if any. The default mode is Auto.

UMA_NONE — no UMA memory will be allocated.
UMA_SPECIFIED — up to the requested UMA memory will be allocated.

UMA_AUTO — allocate the optimum UMA memory size for the platform. For fusion platforms this will provide the optimum UMA allocation for the platform and for other platforms will be the same as NONE.

BLDCFG_UMA_ALLOCATION_SIZE

Provide a build time customization of the UMA allocation size input. If the UmaAllocationMode is SPECIFIED, up to this amount of UMA memory will be allocated, otherwise, it will be ignored. The size must be a multiple of 64 KBytes.

BLDCFG_LIMIT_MEMORY_TO_BELOW_1TB

This item provides for ensuring that the top of memory is limited to below 1 TByte. This may be needed for certain operating systems.

FALSE - Not limited. Top of memory will be greater than 1 TByte if sufficient memory is installed. Memory may be hoisted above 1 TByte to make it accessible.
TRUE - Limited (default). Top of memory will be below 1 TByte no matter how much memory is installed; any additional memory will not be accessible.

BLDCFG_ENABLE_EXTERNAL_VREF_FEATURE

This item specifies whether to use an external, platform specific, memory Vref control.

FALSE - Use internal Vref control (default).
TRUE - Use the callout AgesaExternal2dTrainVrefChange.

BLDCFG_DIMM_TYPE_USED_IN_MIXED_CONFIG

Specifies how to handle installations of mixed memory technology. If the processor supports more than one memory technology, only channels with the specified memory technology will be enabled and other channels will be disabled. Both sets of channels must contain supported memory technology DIMMs. Specifying a memory type that is not supported by the processor is treated the same as specifying UNSUPPORTED_TECHNOLOGY. The default is DDR3_TECHNOLOGY if the processor supports it, otherwise it is family specific.

DDR2_TECHNOLOGY - Unsupported.
DDR3_TECHNOLOGY - Enable only DDR3 memory channels (default).
GDDR5_TECHNOLOGY - Enable only GDDR5 memory channels.
UNSUPPORTED_TECHNOLOGY - Do not support mixed installations.

BLDCFG_PMU_TRAINING_MODE

For processors which provide a Phy Micro-controller Unit (PMU), this parameter selects the memory data bus training mode which the PMU will perform.

PMU_TRAIN_1D - perform 1D training only.
PMU_TRAIN_1D_2D_READ - perform both 1D and 2D read only training.
PMU_TRAIN_1D_2D - perform both 1D and 2D training.
PMU_TRAIN_AUTO - The training mode will be automatically selected based on configuration (default).

BLDCFG_HEAP_DRAM_ADDRESS

During the transition from pre-memory use of the cache as RAM to the post memory initialization time when main memory storage becomes available, the contents of the AGESA™ software heap must be transferred. This configuration element allows the platform to specify the main memory address where the heap contents will be temporarily stored during the transition. The default memory address used is 0x000B0000. The temporary storage location must be 64K in length. This location must be in the 32bit address space and must be preserved after the call to AmdInitPost until the call to AmdInitEnv.

Example:

```
#define BLCFG_HEAP_DRAM_ADDRESS 0x00600000
```

16.2.4 FCH Elements

BLDCFG_SMBUS0_BASE_ADDRESS

This item specifies a customized base I/O address for the processor's System Management Bus (SMBUS) primary controller. The default is an AMD assigned address.

BLDCFG_SMBUS1_BASE_ADDRESS

This item specifies a customized base I/O address for the processor's System Management Bus (SMBUS) secondary controller. The default is an AMD assigned address.

BLDCFG_SIO_PME_BASE_ADDRESS

This item specifies a customized base I/O address for the System I/O (SIO) controller Power Management Event (PME) block. The default is an AMD assigned address.

BLDCFG_ACPI_PM1_EVT_BLOCK_ADDRESS

This item specifies a customized base I/O address for the processor's ACPI PM1 event block. The default is an AMD assigned address.

BLDCFG_ACPI_PM1_CNT_BLOCK_ADDRESS

This item specifies a customized base I/O address for the processor's ACPI PM1 control block. The default is an AMD assigned address.

BLDCFG_ACPI_PM_TMR_BLOCK_ADDRESS

This item specifies a customized base I/O address for the processor's ACPI Power Management Timer block. The default is an AMD assigned address.

BLDCFG_ACPI_CPU_CNT_BLOCK_ADDRESS

This item specifies a customized base I/O address for the processor's ACPI CPU control register block. The default is an AMD assigned address.

BLDCFG_ACPI_GPE0_BLOCK_ADDRESS

This item specifies a customized base I/O address for the processor's General Purpose Event (GPE) block. The default is an AMD assigned address.

BLDCFG_WATCHDOG_TIMER_BASE

This item specifies a customized base address for the processor's Watch Dog Timer. The default is an AMD assigned address.

BLDCFG_ACPI_PMA_BLK_ADDRESS

This item specifies a customized base I/O address for the processor's ACPI PMA power management controller. The default is an AMD assigned address.

BLDCFG_SMI_CMD_PORT_ADDRESS

This item specifies a customized base I/O address for the processor's SMI command port. The default is an AMD assigned address.

BLDCFG_ROM_BASE_ADDRESS

This item specifies a customized base address for the system SPI ROM. The default is an AMD assigned address.

BLDCFG_GEC_SHADOW_ROM_BASE

This item specifies a customized base address for the GEC Shadow ROM. The default is an AMD assigned address.

BLDCFG_HPET_BASE_ADDRESS

This item specifies a customized base address for the processor's High Precision Event Timer (HPET). The default is an AMD assigned address.

BLDCFG_AZALIA_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's Azalia HD audio controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_SMBUS_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's System Management Bus (SMBUS) controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_IDE_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's IDE Controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_SATA_AHCI_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's SATA AHCI controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_SATA_IDE_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's SATA IDE controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_SATA_RAID5_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's SATA RAID 5 Controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_SATA_RAID_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's SATA RAID controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_EHCI_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's EHCI USB controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_OHCI_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's OHCI USB controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_LPC_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's LPC controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_SD_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's Secure Digital (SD) controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_XHCI_SSID

This item specifies a customized PCIe® Subsystem ID for the processor's XHCI controller. The default is an AMD assigned device ID with the AMD vendor ID.

BLDCFG_FCH_PORT80_BEHIND_PCIB

This item specifies whether port 80 access should be directed to PCIe® or LPC.

TRUE - direct port 80 accesses to PCIe®.
FALSE - direct port 80 accesses to LPC (default).

BLDCFG_FCH_ENABLE_ACPI_SLEEP_TRAP

This item specifies whether SMI traps are enabled for ACPI sleep.

TRUE - ACPI sleep SMI traps are enabled (default).
FALSE - ACPI sleep SMI traps are disabled.

BLDCFG_FCH_GPP_LINK_CONFIG

This item allows the GPP Link configuration to be specified to match the limits of the board or to balance performance.

0 - Port A has all four lanes (default).
2 - Port A has two lanes and Port B has two lanes.
3 - Port A has two lanes and Ports B and C each have one lane.
4 - All four ports have one lane each.

BLDCFG_FCH_GPP_PORT0_PRESENT

This item indicates whether GPP Port 0 is present.

FALSE - The port is not present (default).
TRUE - The port is present.

BLDCFG_FCH_GPP_PORT1_PRESENT

This item indicates whether GPP Port 1 is present.

FALSE - The port is not present (default).
TRUE - The port is present.

BLDCFG_FCH_GPP_PORT2_PRESENT

This item indicates whether GPP Port 2 is present.

FALSE - The port is not present (default).
TRUE - The port is present.

BLDCFG_FCH_GPP_PORT3_PRESENT

This item indicates whether GPP Port 3 is present.

FALSE - The port is not present (default).
TRUE - The port is present.

BLDCFG_FCH_GPP_PORT0_HOTPLUG

This item indicates whether GPP Port 0 supports hotplug.

FALSE - Hotplug is not supported by the port (default).
TRUE - Hotplug is supported by the port.

BLDCFG_FCH_GPP_PORT1_HOTPLUG

This item indicates whether GPP Port 1 supports hotplug.

FALSE - Hotplug is not supported by the port (default).
TRUE - Hotplug is supported by the port.

BLDCFG_FCH_GPP_PORT2_HOTPLUG

This item indicates whether GPP Port 2 supports hotplug.

FALSE - Hotplug is not supported by the port (default).
TRUE - Hotplug is supported by the port.

BLDCFG_FCH_GPP_PORT3_HOTPLUG

This item indicates whether GPP Port 3 supports hotplug.

FALSE - Hotplug is not supported by the port (default).
TRUE - Hotplug is supported by the port.

16.3 Customizing the Environment—Library Functions

These procedures interface the application software with the hardware. Most of these procedures provide access to processor instructions not available in the standard C language. These procedures are documented for the expert user who needs to modify the underpinnings of the software for their platform configuration. This is useful when porting the software to new software code bases.

Procedures are listed in alphabetical order.

WARNING: Only the published entry points are assured to be available over time. Use of other procedure names by the host environment that were found by code inspection is strongly discouraged. AMD reserves the right to modify or remove internal procedure names not published in this specification.

Common Parameters

All of the environment library functions accept the following parameter:

StdHeader

Pointer to a StdHeader. This structure was defined in “Standard Header” on page 32 and contains host environment data that the library function can use in performing its function.

Similarly, many library functions use the operation width-definition type:

```
typedef enum ACCESS_WIDTH {
  AccessWidth8 = 1,
  AccessWidth16,
  AccessWidth32,
  AccessWidth64,
} ACCESS_WIDTH;
```

The complete definition can be found in the AGESA.H include file.

LibAmdCpuidRead

Perform the CPUID instruction and return the response.

Prototype

```
VOID LibAmdCpuidRead (
     IN      UINT32            CpuidFcnAddress,
     OUT     CPUID_DATA        *Value,
     IN OUT  AMD_CONFIG_PARAMS *StdHeader
);
```

Parameters

CpuidFcnAddress

Parameter to the CPUID instruction passed in EAX.

Value

Pointer to location where to place the instruction response.

Related Definitions

```
typedef struct {
     OUT     UINT32      EAX_Reg;
     OUT     UINT32      EBX_Reg;
     OUT     UINT32      ECX_Reg;
     OUT     UINT32      EDX_Reg;
} CPUID_DATA;
```

Description

This library function executes the processor CPUID instruction and stores the response values into the provided data structure.

LibAmdIoRead
LibAmdIoWrite

Perform an Input/Output cycle using the IN or OUT processor instruction.

Prototype

```
VOID LibAmdIoRead (
    IN      ACCESS_WIDTH      AccessWidth,
    IN      UINT16            IoAddress,
    OUT     VOID              *Value,
    IN OUT  AMD_CONFIG_PARAMS *StdHeader
);

VOID LibAmdIoWrite (
    IN      ACCESS_WIDTH      AccessWidth,
    IN      UINT16            IoAddress,
    IN      VOID              *Value,
    IN OUT  AMD_CONFIG_PARAMS *StdHeader
);
```

Parameters

AccessWidth

Indicates the data size of the operation.

IoAddress

Address in the IO address space to which the operation pertains.

Value

Pointer to location where to obtain (Write) or place (Read) the instruction data value.

Description

Performs the IO cycle of the specified size to the indicated IO space address.

LibAmdLocateImage

Find the binary image containing a module with the specified signature.

Prototype

```
VOID *LibAmdLocateImage (
    IN      VOID              *StartAddress,
    IN      VOID              *EndAddress,
    IN      UINT32            Alignment,
    IN      CHAR8             ModuleSignature[8]
);
```

Parameters

StartAddress

Pointer to location where to start the search.

EndAddress

Pointer to location where to end the search.

Alignment

Specifies how big a step to take when searching. If this value is zero, then a default value of 4096 bytes is used.

ModuleSignature

Pointer to an 8-character string that matches the module identifier.

Description

This procedure searches from *StartAddress* to *EndAddress* incrementing, or stepping, by *Alignment* bytes to locate a code module with the identifier that matches the *ModuleSignature* parameter.

Returned Value

The functions return value is a pointer to the binary image containing the module. If the return value is NULL, then a module with the specified signature could not be located.

LibAmdMemCopy
LibAmdMemFill

Move or fill a block of memory.

Prototype

```
VOID LibAmdMemCopy (
    IN      VOID        *Destination,
    IN      VOID        *Source,
    IN      UINTN       CopyLength,
    IN OUT  AMD_CONFIG_PARAMS *StdHeader
);

VOID LibAmdMemFill (
    IN      VOID        *Destination,
    IN      UINT8       Value,
    IN      UINTN       FillLength,
    IN OUT  AMD_CONFIG_PARAMS *StdHeader
);
```

Parameters

Source

Pointer to the memory location where to find the data.

Destination

Pointer to the memory location where to place the data.

CopyLength

FillLength

Length in bytes of the memory area to be copied or filled.

Value

Value with which to fill the memory area.

Description

The MemCopy procedure copies data from one memory location to another. The data is copied from low address to high address. Overlapping source and destination areas must have the destination area begin at least 4 bytes higher than the originating area.

The MemFill procedure fills the memory area with the value provided.

LibAmdMemRead
LibAmdMemWrite

Access a memory location above the 4-Gbyte address range.

Prototype

```
VOID LibAmdMemRead (
     IN      ACCESS_WIDTH      AccessWidth,
     IN      UINT64            MemAddress,
     OUT     VOID              *Value,
     IN OUT  AMD_CONFIG_PARAMS *StdHeader
);

VOID LibAmdMemWrite (
     IN      ACCESS_WIDTH      AccessWidth,
     IN      UINT64            MemAddress,
     IN      VOID              *Value,
     IN OUT  AMD_CONFIG_PARAMS *StdHeader
);
```

Parameters

AccessWidth

Indicates the data size of the operation.

MemAddress

Address where to perform the operation.

Value

Pointer to location where to obtain (Write) or place (Read) the operation data value.

Description

These procedures use a special access mechanism of the AMD processor to read or write a value to a memory location above the 4-Gbyte address boundary—an address needing more than 32 bits to represent, thus a standard C language pointer could not access. The calling procedure is responsible to assure that the data location is of the proper size or type to contain the data.

LibAmdMsrRead
LibAmdMsrWrite

Perform an RDMSR or WRMSR processor instruction and return the response.

Prototype

```
VOID LibAmdMsrRead (
    IN      UINT32            MsrAddress,
    OUT     UINT64            *Value,
    IN OUT  AMD_CONFIG_PARAMS *StdHeader
);

VOID LibAmdMsrWrite (
    IN      UINT32            MsrAddress,
    IN      UINT64            *Value,
    IN OUT  AMD_CONFIG_PARAMS *StdHeader
);
```

Parameters

MsrAddress

MSR register number or address where to perform the operation.

Value

Pointer to location where to obtain (Write) or place (Read) the operation data value.

Description

These library functions execute the processor RDMSR or WRMSR instruction. The MsrRead procedure accesses the MSR register and stores the contents into the provided data location. The MsrWrite sets the contents of the MSR to the value in the provided data location.

LibAmdPciRead
LibAmdPciWrite

Generate a PCI configuration cycle to access a PCI device register.

Prototype

```
VOID LibAmdPciRead (
     IN      ACCESS_WIDTH      AccessWidth,
     IN      PCI_ADDR          PciAddress,
     OUT     VOID              *Value,
     IN OUT  AMD_CONFIG_PARAMS *StdHeader
);

VOID LibAmdPciWrite (
     IN      ACCESS_WIDTH      AccessWidth,
     IN      PCI_ADDR          PciAddress,
     IN      VOID              *Value,
     IN OUT  AMD_CONFIG_PARAMS *StdHeader
);
```

Parameters

AccessWidth

Indicates the data size of the operation.

PciAddress

The PCI device register address where to perform the operation.

Value

Pointer to location where to obtain (Write) or place (Read) the operation data value.

Related Definitions

```
typedef struct {
     IN OUT  UINT32      Register:12;
     IN OUT  UINT32      Function:3;
     IN OUT  UINT32      Device:5;
     IN OUT  UINT32      Bus:8;
     IN OUT  UINT32      Segment:4;
} EXT_PCI_ADDR;
```

Description

These procedures use the bit field PCI register address to locate the target register and then perform a PCI configuration access cycle to read or write the target register. If the PcieBasePtr element of the StdHeader is filled with a proper pointer (not = NULL) to a memory-mapped

IO region, then the access is performed through the MMIO base, else the access is performed using the PCI configuration IO ports.

Section V - Integrated Debug Services (IDS)

Chapter 17 IDS Overview

The Integrated Debug Services (IDS) of the AGESA™ software bring several abilities to the host environment and the platform developer that enhance the ability to alter code actions and give flexibility of choice to the platform developer. As indicated by the name, this sub-system is intended for use in debugging a platform in preparing it for production delivery.

The intent is for these controls to be removed for the production build.

17.1 Goals

The IDS sub-system is intended to aid platform debugging and, optionally, provide a mechanism to import end-user choices into the operations of the platform. The goals of this sub-system are:

- Provide Controls structure to allow flexibility in defining the platform feature set operation
- Create a common feature control set across all platforms
- Be closely integrated with the AGESA™ software, providing a consistent interface over time
- Define a common data definition for interfacing to the host environment
- Implement entirely in the C language
- Optional inclusion—because these controls are not intended for the production build, the controls that are not included in the build implementation have a Zero code-size impact.

17.2 Capabilities

The IDS sub-system provides services in three broad categories:

Configuration Controls - The IDS controls allow a porting engineer to make run time decisions which affect configuration settings in the system. These controls typically select an alternative value for a configuration register bitfield for the purposes of testing the hardware platform during the ‘bring-up’ phase. This is the period a platform manufacturer is readying the platform for production.

Trace services - The IDS sub-system provides several tools to trace the progression of certain algorithms in the AGESA™ software. These tools can give insight to the platform engineer about what is happening on the platform.

Performance Analysis - The IDS sub-system has the capability to collect time stamp data at certain points in the code. This data can then be analyzed to show what sections of code are taking more time than others

17.3 Host Environment

The Integrated Debug Services rely upon having access to run time decisions made by the platform developer. The results of these decisions are recorded in non-volatile storage within the platform.

Figure 17.2 gives an overview of the host environment and the interaction with the IDS. The figure shows two distinct sections:

The "**front end UI**" - This is the user interface (UI) that enables the platform developer make decisions and stores information about those decisions in non-volatile storage (NV). This code and services exist in and are provided by the host environment. For example, the SETUP utility. Each BIOS vendor supplies a version of SETUP. (The UEFI environment uses a standard UI tool. For the UEFI environment AMD supplies code components to insert into the standard UI tool, which provides the user selection ability for the IDS control fields defined in this specification.)

The "**back end code**" - This is the code embedded into the AGESA™ software. This code invokes actions based upon the decisions made by the platform developer.

The communication method between these two sections is based on an "IDS element ID." Each piece of data used by the IDS is identified by a unique ID value, the 'element ID'. The element ID names are defined in the include file AGESA.h.

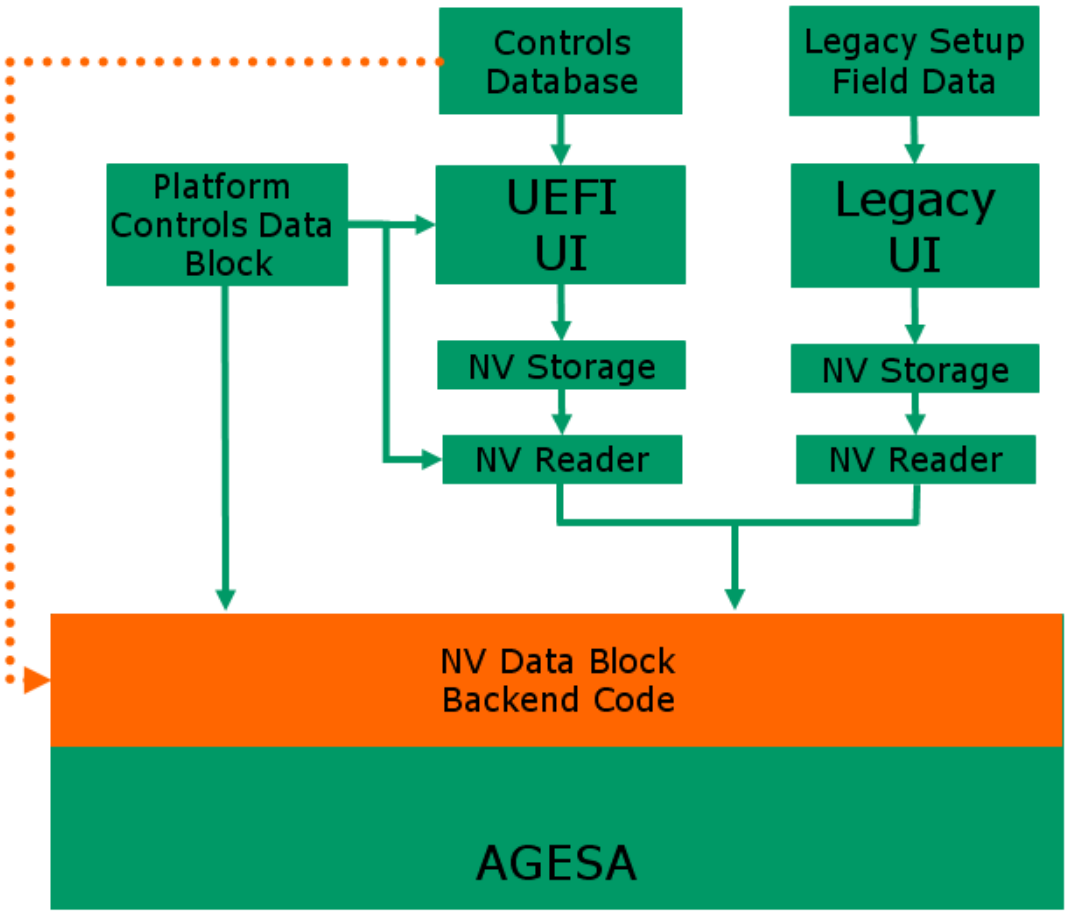

Figure 17.2: IDS overview

The back-end code uses the call-out "AgesaGetIdsData" on page 162 to obtain values for the IDS elements.

17.4 Installation and Configuration

Because the IDS is intended for advanced to expert users for the purpose of debugging a platform for production delivery, the reader and user of these features are assumed to have an advanced to expert level of programming ability.

The IDS sub-system uses several build switches to allow the platform developer to specify how many and which IDS controls are enabled in the back-end code. They allow the platform developer to control the trade-off between flexibility, code size, and boot time.

The IDS build switches are defined in a special file named OptionsIds.h located in the platform build tip directory. An example version of this file is available in 20.2 “Example” on page 254. The platform developer may copy this example to the platform build tip directory and make changes to enable the IDS support desired.

The build switches which are available are defined along with the configuration controls they affect in “IDS Build Switches” on page 246.

Chapter 18 IDS Macros

Throughout the core files there are macros which import debug services. These are the Integrated Debug Services (IDS) macros further defined in “IDS Configuration Controls” on page 246. Expansion of all IDS macros is controlled by #define compile switches defined for the platform. When disabled, the macros are not expanded during compile and consume no code space.

These macros are used to establish execution hooks in the AGESA™ software so that the special services can be invoked.

AGESA_TESTPOINT

Prototype

```
AGESA_TESTPOINT(TestPoint)
```

Parameters

TestPoint

The value for display indicating progress

Description

This Macro is always enabled. The default action is to write the TestPoint value to an I/O port. The I/O port is 8 bits in size and the default address is 0x80. IDS_DEBUG_PORT may be defined to another value to use a different address. Example:

```
#define IDS_DEBUG_PORT 0x84
```

STOP_HERE

Prototype

```
STOP_HERE(VOID)
```

Parameters

<none>

Description

This macro is always enabled. The default action is to generate a compiler error.

This macro forces a program halt to allow a debugger tool to gain control and investigate the platform. This macro should NOT remain in the code after the debug session is complete, thus the default action is to generate a compiler error to flag the developer to remove the macro.

Dependencies

If IDSOPT_ASSERT_ENABLED is TRUE, then this macro causes the program to halt with stop code display (see “Debugging Using IDS” on page 255).

ASSERT

Prototype

```
ASSERT(*expression*)
```

Parameters

expression

This is an expression that must evaluate to a BOOLEAN value.

Description

This macro makes the assertion that *expression* is TRUE. This is used to do parameter checks, bounds checking, range checks, and "sanity" checks.

Dependencies

If IDSOPT_ASSERT_ENABLED is TRUE, then this macro evaluates the `*expression*` to a BOOLEAN value. If the `*expression*` is true, then no action is taken (no error). If the `*expression*` is False, an error stop is generated to halt the program with stop code display (see "Debugging Using IDS" on page 255). Use this to check for software errors that must be resolved before production.

IDS_ERROR_TRAP

Prototype

```
IDS_ERROR_TRAP;
```

Parameters

There are no parameters to IDS_ERROR_TRAP.

Description

This macro works similarly to a use of “ASSERT (FALSE);”. This is used to provide debug support for severe errors, including those potentially caused by wrapper and platform BIOS errors. These severe errors are normally reported to the host envirnomment via an AGESA return status. This macro can be enabled to debug the cause closer to the source of the error. The macro is used for cases that return AGESA_ERROR, AGESA_CRITICAL, and AGESA_FATAL.

Dependencies

If IDSOPT_ERROR_TRAP_ENABLED is TRUE, then this macro generates an error stop to halt the program with stop code display (see “Debugging Using IDS” on page 255).

IDS_OPTION_HOOK

Prototype

```
IDS_OPTION_HOOK(IdsOption, DataPtr, StdHeader);
```

Parameters

IdsOption

IDS Option index to select the function to perform at this hook point.

DataPtr

Pointer to data relevant to the hook point. This may contain values the hook point procedure overrides.

StdHeader

Pointer to the standard header.

Description

Invoke the specified IDS option hook procedure. These hook procedures may perform actions to modify an algorithm or make changes to an entry point's input parameters or modify specific register values. In general, an option hook adds code to the code seen in the source file.

Dependencies

If IDSOPT_IDS_ENABLED is TRUE, then this macro calls the specified IDS hook procedure, passing the DataPtr and StdHeader as parameters.

IDS_SKIP_HOOK

Prototype:

```
IDS_SKIP_HOOK(IdsOption, DataPtr, StdHeader) {
     // skipable code
};
```

Parameters

IdsOption

IDS Option index to select the function to perform at this hook point. This function returns a boolean value.

DataPtr

Pointer to data relevant to the hook point. This may contain values the hook point procedure overrides.

StdHeader

Pointer to the standard header.

Description

Invoke the specified IDS skip hook procedure. In general, a skip hook removes code from execution from what is seen in the source file. The IdsOption function returns a boolean. The macro evaluates the boolean and if it is FALSE, it skips the code inside the curly braces. If the boolean is TRUE then the code inside the curly braces is executed.

These hook procedures may also perform actions to modify an algorithm or make changes to an entry point's input parameters or modify specific register values. Another potential use is to replace the main code inside the curly braces with an alternate debug version.

Dependencies

If IDSOPT_IDS_ENABLED is TRUE, then this macro calls the specified IDS hook procedure, passing the DataPtr and StdHeader as parameters. If IDSOPT_IDS_ENABLED is FALSE, then the macro is not expanded and the code inside the curly braces is always executed.

IDS_HDT_CONSOLE

Prototype:

```
IDS_HDT_CONSOLE(Group, FormatString, <VarArgs...>)
```

Parameters

Group

Specifies which group this debug output belongs to, so that it can be filtered by the HDT script.

FormatString

Text string to be displayed with an HDT script. This string uses classic 'printf' format to display the arguments that comes after it.

VarArgs

List of arguments that will be formatted and displayed along with FormatString.

Description

This macro invokes communications with the HDT to allow users to pipe out trace data from BIOS. The protocol of this communication is described in the IDS Doxygen internal documentation. Users can write their own HDT console script to extract data or use the AGESA™ provided scripts supplied in the AMDTools sub-directory.

Dependencies

Ensure $(AGESA_OptsDir)\OptionsIds.h exists and contains these two definitions:

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_TRACING_ENABLED must be TRUE.

See "HDT Console" on page 256 for additional setup and use information.

CONSOLE

Prototype:

```
CONSOLE(FormatString, <VarArgs...>)
```

Parameters

FormatString

Text string to be displayed with an HDT script. This string uses classic ‘printf’ format to display the arguments that comes after it.

VarArgs

List of arguments that will be formatted and displayed along with FormatString.

Description

This macro is always enabled. The default action is to generate a compiler error.

This macro invokes communications with the HDT to allow users to pipe out trace data from BIOS. The protocol of this communication is described in the IDS Doxygen internal documentation. Users can write their own HDT console script to extract data or use the AGESA™ provided scripts supplied in the AMDTools sub-directory. This macro should NOT remain in the code after the debug session is complete, thus the default action is to generate a compiler error to flag the developer to remove the macro.

Dependencies

Ensure $(AGESA_OptsDir)\OptionsIds.h exists and contains these two definitions:

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_TRACING_ENABLED must be TRUE.

See “HDT Console” on page 256 for additional setup and use information.

IDS_HDT_CONSOLE_INIT

Prototype:

```
IDS_HDT_CONSOLE_INIT(AMD_CONFIG_PARAMS *StdHeader)
```

Parameters

StdHeader

Pointer to a standard header structure.

Description

This macro is used to initiate communications with the HDT console script.

Dependencies

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_TRACING_ENABLED must be TRUE.

See “HDT Console” on page 256 for additional setup and use information.

IDS_HDT_CONSOLE_EXIT

Prototype:

```
IDS_HDTOUT_EXIT(AMD_CONFIG_PARAMS *StdHeader)
```

Parameters

StdHeader

Pointer to a standard header structure.

Description

This macro closes the communication session with the HDT script.

Dependencies

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_TRACING_ENABLED must be TRUE.

See “HDT Console” on page 256 for additional setup and use information.

IDS_PERF_TIMESTAMP

Prototype:

```
IDS_PERF_TIMESTAMP(AMD_CONFIG_PARAMS *StdHeader, UINT32 TestPoint)
```

Parameters

StdHeader

Pointer to a standard header structure.

TestPoint

The AGESA software test point value for which to record this time stamp.

Description

This macro records the time stamp counter for use later to analyze boot times.

Dependencies

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_PERF_ANALYSIS must be TRUE.

IDS_PERF_ANALYSE

Prototype:

```
IDS_PERF_ANALYSE(AMD_CONFIG_PARAMS *StdHeader)
```

Parameters

StdHeader

Pointer to a standard header structure.

Description

This macro transfers the time stamp data to the HDT script for analysis.

Dependencies

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_PERF_ANALYSIS must be TRUE.

IDS_EXCEPTION_TRAP

Prototype:

```
IDS_EXCEPTION_TRAP(FunctionId, DataPtr, StdHeader)
```

Parameters

FunctionId

Designates the action to be taken.

IDS_IDT_REPLACE_IDTR_FOR_BSC - Create an IDT for the boot core, saving any IDT currently in use.
IDS_IDT_RESTORE_IDTR_FOR_BSC - Restore any IDT that was previously in use for the boot core.
IDS_IDT_UPDATE_EXCEPTION_VECTOR_FOR_AP - Update the exception vectors.

DataPtr

Provide data relevant to the desired action. For updating vectors, provide the base address of the IDT.

StdHeader

Pointer to a standard header structure.

Description

This macro installs an IDT for catching exceptions. All the exception handlers are set to a spin loop routine. During debug, this prevents faults or resets and allows analysis of why the exception occurred.

Dependencies

Ensure $(AGESA_OptsDir)\OptionsIds.h exists and contains these two definitions:

IDSOPT_IDS_ENABLED must be TRUE.

IDSOPT_IDT_EXCEPTION_TRAP must be TRUE.

Chapter 19 IDS Configuration Controls

19.1 IDS Build Switches

The IDS enable switches are layered and grouped to allow fine tuning of the enabled pieces.

The defined IDS build switches are listed below. Each option switch is a BOOLEAN, so can be set to either TRUE or FALSE. The default setting is FALSE.

IDSOPT_IDS_ENABLED

This is the master switch for the IDS sub-system. Use this option to enable or remove the entire IDS feature set. This switch must be TRUE for any of the other options to function.

IDSOPT_ASSERT_ENABLED

This is the switch for halting support for debugging. If the switch is TRUE ASSERT macros will be enabled. Also temporary use of STOP_HERE is enabled when this switch is TRUE.

IDSOPT_ERROR_TRAP_ENABLED

This switch enables additional debug halting associated with detected severe events.

IDSOPT_CONTROL_ENABLED

This is the main switch for the IDS configuration controls. This switch must be TRUE for any of the configuration controls to function.

IDSOPT_HEAP_CHECKING

When set to TRUE this switch enables internal checking of heap consistency, checking for issues such as buffer overrun.

IDSOPT_TRACING_ENABLED

This is the main switch for the IDS console controls, macros and support code needed to enable tracing of algorithms. This switch must be TRUE for any of the tracing features to function.

IDSOPT_TRACE_USER_OPTIONS

This switch enables reporting of the user options on the console. IDSOPT_TRACING_ENABLED must be TRUE for this switch to have any effect. Code size increases due to enabling tracing can be reduced by setting this switch to FALSE.

IDSOPT_IDT_EXCEPTION_TRAP

When set to TRUE, this switch enables use of IDS_EXCEPTION_TRAP to debug exceptions.

IDSOPT_PERF_ANALYSIS

This is the main switch for the IDS performance analysis controls, macros and support code needed to enable time data gathering. This switch must be TRUE for any of the performance analysis features to function.

19.2 UI Interface Controls

The controls in this section are enabled by the *IDSOPT_CONTROL_ENABLED* build option switch.

These User Interface (UI) controls allow the Platform developer to adjust a limited set configuration registers or enable diagnostic tools for the purpose of debugging their platform hardware. These controls are intended for use in the platform 'bring-up' and debug phase of preparing the platform for production release. The IDS sub-system must be disabled for the production release.

The IDS configuration controls exist because the affected settings are not exposed through the defined program interfaces. The affected settings will be platform constants or automatically determined by the AGESA™ software in the production release.

The following sections describe the user selection controls that are provided and the selection option values available for those controls. These will appear in the following format:

AGESA_IDS_NV_CONTROL_NAME (This is the identifier name defined in the IDS.H file)

This control does this.... *(this is the description of the control)*

IDS_SELECTION	Value description. *(Each valid selection is listed by the identifier name defined in the IDS.H file.)*
IDS_SELECTION	Value description.

Please note that when activated, the IDS control overrides any corresponding build configuration option that may affect the same setting. There are no default values, the value returned from the IDS user interface will be the value used. It is the responsibility of the host environment user interface to provide any user default settings and assure the final selection value is proper for the control field.

19.2.1 Configuration Controls

19.2.1.1 Processor Core Controls

AGESA_IDS_NV_UCODE

This control provides the ability to control whether the microcode patch will be loaded. This control will be evaluated at the entry point AmdInitReset and will override the default action of loading microcode patches.

IDS_DISABLED	Disable the microcode patching. No patch will be loaded even if one is present that matches the processor.
IDS_ENABLED	Enable the microcode patching. Normal patch loading action per the BKDG.

AGESA_IDS_NV_TARGET_PSTATE

This control provides the ability to specify the P-state at which entry to OS will occur. This control will be evaluated at the entry point AmdInitLate. This control may be useful for an OS that does not support ACPI.

IDS_TARGETPSTATE_AUTO	Automatic. No override is performed. The PState is determined by BIOS based on the installed processors' capabilities.
IDS_TARGETPSTATE_HIGHEST	set the target PState to highest
IDS_TARGETPSTATE_LOWEST	set the target PState to Lowest
IDS_TARGETPSTATE_0	set the target PState to state 0 if supported
IDS_TARGETPSTATE_1	set the target PState to state 1 if supported
IDS_TARGETPSTATE_2	set the target PState to state 2 if supported
IDS_TARGETPSTATE_3	set the target PState to state 3 if supported
IDS_TARGETPSTATE_4	set the target PState to state 4 if supported
IDS_TARGETPSTATE_5	set the target PState to state 5 if supported
IDS_TARGETPSTATE_6	set the target PState to state 6 if supported
IDS_TARGETPSTATE_7	set the target PState to state 7 if supported

AGESA_IDS_NV_POSTPSTATE

This control provides the ability to override the P-state entered before memory initialization during BIOS post. This control will be evaluated at the entry point AmdInitEarly.

IDS_POSTPSTATE_AUTO	Automatic base on BIOS execution
IDS_POSTPSTATE_HIGHEST	set the post PState to highest
IDS_POSTPSTATE_LOWEST	set the post PState to Lowest
IDS_POSTPSTATE_0	set the post PState to state 0 if supported
IDS_POSTPSTATE_1	set the post PState to state 1 if supported
IDS_POSTPSTATE_2	set the post PState to state 2 if supported
IDS_POSTPSTATE_3	set the post PState to state 3 if supported
IDS_POSTPSTATE_4	set the post PState to state 4 if supported
IDS_POSTPSTATE_5	set the post PState to state 5 if supported
IDS_POSTPSTATE_6	set the post PState to state 6 if supported
IDS_POSTPSTATE_7	set the post PState to state 7 if supported

Note that if the desired P-State number is numerically greater than the lowest performance P-State, the core is set to the lowest performance P-State.

AGESA_IDS_NV_POWER_POLICY

This control provides the ability to override settings which provide a tradeoff of power savings or performance. See the internal documentation, described in "Internal Documentation" on page 25, for details on what settings are affected by this control.

IDS_POWER_POLICY_AUTO Policy is set according to configuration.
IDS_POWER_POLICY_POWER Policy is forced to maximize battery life.
IDS_POWER_POLICY_PERFORMANCE Policy is forced to maximize performance.

19.2.1.2 Memory Cache & ECC Controls

AGESA_IDS_NV_SCRUB_REDIRECTION

This control provides the ability to override build parameters used by the memory routines. This control will be evaluated at the entry point AmdInitPost, and used as the value for the CfgEccRedirection parameter.

IDS_DISABLED Disable this feature.
IDS_ENABLED Enable the northbridge to force a write to dram with corrected data when a correctable error on the dram bus is detected during a normal CPU or bus master read request.

See Build configuration item "BLDCFG_ECC_REDIRECTION" on page 211.

AGESA_IDS_NV_DRAM_SCRUB

This control provides the ability to override build parameters used by the memory routines. This control will be evaluated at the entry point AmdInitPost, and used as value of the CfgScrubDramRate parameter. Below are the options used to set the rate of background scrubbing for DRAM.

IDS_DRAMSCRUB_DISABLED	IDS_DRAMSCRUB_20_0NS
IDS_DRAMSCRUB_40NS	IDS_DRAMSCRUB_80NS
IDS_DRAMSCRUB_160NS	IDS_DRAMSCRUB_320NS
IDS_DRAMSCRUB_640NS	IDS_DRAMSCRUB_1_28US
IDS_DRAMSCRUB_2_56US	IDS_DRAMSCRUB_5_12US
IDS_DRAMSCRUB_10_2US	IDS_DRAMSCRUB_20_5US
IDS_DRAMSCRUB_41_0US	IDS_DRAMSCRUB_81_9US
IDS_DRAMSCRUB_163_8US	IDS_DRAMSCRUB_327_7US
IDS_DRAMSCRUB_655_4US	IDS_DRAMSCRUB_1_31MS
IDS_DRAMSCRUB_2_62MS	IDS_DRAMSCRUB_5_24MS
IDS_DRAMSCRUB_10_49MS	IDS_DRAMSCRUB_20_97MS
IDS_DRAMSCRUB_42_0MS	IDS_DRAMSCRUB_84_0MS

See Build configuration item "BLDCFG_SCRUB_DRAM_RATE" on page 211.

AGESA_IDS_NV_DCACHE_SCRUB

This control provides the ability to override build parameters used by the memory routines. This control will be evaluated at the entry point AmdInitPost, and used as the value for the CfgScrubDcRate parameter. This option is used to set the rate of background scrubbing for the DCache.

IDS_DCACHESCRUB_DISABLED	IDS_DCACHESCRUB_40NS
IDS_DCACHESCRUB_80NS	IDS_DCACHESCRUB_160NS
IDS_DCACHESCRUB_320NS	IDS_DCACHESCRUB_640NS
IDS_DCACHESCRUB_1_28US	IDS_DCACHESCRUB_2_56US
IDS_DCACHESCRUB_5_12US	IDS_DCACHESCRUB_10_2US
IDS_DCACHESCRUB_20_5US	IDS_DCACHESCRUB_41_0US
IDS_DCACHESCRUB_81_9US	IDS_DCACHESCRUB_163_8US
IDS_DCACHESCRUB_327_7US	IDS_DCACHESCRUB_655_4US
IDS_DCACHESCRUB_1_31MS	IDS_DCACHESCRUB_2_62MS
IDS_DCACHESCRUB_5_24MS	IDS_DCACHESCRUB_10_49MS
IDS_DCACHESCRUB_20_97MS	IDS_DCACHESCRUB_42_0MS
IDS_DCACHESCRUB_84_0MS	

See Build configuration item “BLDCFG_SCRUB_DC_RATE” on page 212.

AGESA_IDS_NV_L2_SCRUB

This control provides the ability to override build parameters used by the memory routines. This control will be evaluated at the entry point AmdInitPost, and used as the value for the CfgScrubL2Rate parameter. This option is used to set the rate of background scrubbing for the L2 cache.

IDS_L2SCRUB_DISABLED	IDS_L2SCRUB_40NS
IDS_L2SCRUB_80NS	IDS_L2SCRUB_160NS
IDS_L2SCRUB_320NS	IDS_L2SCRUB_640NS
IDS_L2SCRUB_1_28US	IDS_L2SCRUB_2_56US
IDS_L2SCRUB_5_12US	IDS_L2SCRUB_10_2US
IDS_L2SCRUB_20_5US	IDS_L2SCRUB_41_0US
IDS_L2SCRUB_81_9US	IDS_L2SCRUB_163_8US
IDS_L2SCRUB_327_7US	IDS_L2SCRUB_655_4US
IDS_L2SCRUB_1_31MS	IDS_L2SCRUB_2_62MS
IDS_L2SCRUB_5_24MS	IDS_L2SCRUB_10_49MS
IDS_L2SCRUB_20_97MS	IDS_L2SCRUB_42_0MS
IDS_L2SCRUB_84_0MS	

See Build configuration item “BLDCFG_SCRUB_L2_RATE” on page 211.

AGESA_IDS_NV_L3_SCRUB

This control provides the ability to override build parameters used by the memory routines. This control will evaluate before entry point AmdInitPost, and used as default value of CfgScrubL3Rate parameter.

May not be available on all families.

See Build configuration item “BLDCFG_SCRUB_L3_RATE” on page 211.

AGESA_IDS_NV_ICACHE_SCRUB

This control provides the ability to override build parameters used by the memory routines. This control will be evaluated at the entry point AmdInitPost, and used as the value for the CfgScrubIcRate parameter.

See Build configuration item “BLDCFG_SCRUB_IC_RATE” on page 212.

AGESA_IDS_NV_SYNC_ON_ECC_ERROR

This control provides the ability to override ECC_OVERRIDE_STRUCT structure which used by MemFInitECC routine.This control will evaluate before entry point AmdInitPost, and used as default value of CfgEccSyncFlood parameter.

IDS_DISABLED Disable this feature.
IDS_ENABLED Enables flooding of all links with sync packets on detection of an uncorrectable ECC error.

See Build configuration item “BLDCFG_ECC_REDIRECTION” on page 211.

AGESA_IDS_NV_ECC_SYMBOL_SIZE

This control provides the ability to override local variable "size" used by memory routines. This control will be evaluated at the entry point AmdInitPost.

IDS_ECCSYMBOLSIZE_X4 x4 symbol size and code used
IDS_ECCSYMBOLSIZE_X8 x8 symbol size and code used

See Build configuration item “BLDCFG_ECC_SYMBOL_SIZE” on page 212.

AGESA_IDS_NV_DRAM_BURST_LENGTH32

This specifies the burst length of DRAM accesses and, as a result, the number of data bytes exchanged in each access

IDS_DRAMBURSTLENGTH32_64BYTE 64 byte mode
IDS_DRAMBURSTLENGTH32_32BYTE 32 byte mode
IDS_DRAMBURSTLENGTH32_AUTO Disable any override. Do no user selection and let the BIOS perform its normal functions.

See Build configuration item “BLDCFG_USE_BURST_MODE” on page 209.

AGESA_IDS_NV_ALL_MEMCLKS

This control provides the ability to override local variable "MemoryAllClocks" used by the memory routines. This control will be evaluated at the entry point AmdInitPost.

IDS_DISABLED Disable this feature
IDS_ENABLED Enables this feature

See Build configuration item “BLDCFG_MEMORY_ALL_CLOCKS_ON” on page 210.

AGESA_IDS_NV_MEMORY_POWER_DOWN_MODE

IDS_PWRDNMODE_CHANNEL Channel CKE control mode.
IDS_PWRDNMODE_CS Chip select CKE control mode.
IDS_PWRDNMODE_AUTO Disable any override. Do no user selection and let the BIOS perform its normal functions.

See Build configuration item “BLDCFG_MEMORY_POWER_DOWN” on page 207.

19.2.2 HDT Tracing controls

The controls in this section are enabled by the *IDSOPT_TRACING_ENABLED* build option switch.

The support code for the control is added to the build when the corresponding build option switches are set for TRUE. In addition, many debug features also have a run-time control to activate the feature.

During the bring-up and debug test period for a platform, the platform developer will want to have the HDT debug services available and built into the executable image, but not necessarily always active, since that would cause extra time in the boot sequence. The following IDS controls are provided to allow the platform developer to select which sections of HDT analysis code are active for the boot sequence.

AGESA_IDS_NV_HDTOUT

This is an IDS control through which the platform developer can turn the tracing capability on or off. The value of this control is boolean and must be TRUE for the tracing macro function to be executed.

19.2.3 Performance Analysis Controls

The controls in this section are enabled by the *IDSOPT_PERF_ANALYSIS* build option switch.

19.3 Build Switch Hierarchy

This sections shows the relationship of the build switches to the macros and controls they enable.

AGESA_TESTPOINT
IDSOPT_IDS_ENABLED
IDS_ASSERT_ENABLED
STOP_HERE

ASSERT
IDS_ERROR_TRAP_ENABLED
IDS_ERROR_TRAP
IDS_OPTION_HOOK
IDS_SKIP_HOOK
IDSOPT_CONTROL_ENABLED
AGESA_IDS_NV_UCODE
AGESA_IDS_NV_TARGET_PSTATE
AGESA_IDS_NV_POSTPSTATE
AGESA_IDS_NV_CORE_LEVEL
AGESA_IDS_NV_SCRUB_REDIRECTION
AGESA_IDS_NV_DRAM_SCRUB
AGESA_IDS_NV_DCACHE_SCRUB
AGESA_IDS_NV_L2_SCRUB
AGESA_IDS_NV_L3_SCRUB
AGESA_IDS_NV_ICACHE_SCRUB
AGESA_IDS_NV_SYNC_ON_ECC_ERROR
AGESA_IDS_NV_ECC_SYMBOL_SIZE
AGESA_IDS_NV_DRAM_BURST_LENGTH32
AGESA_IDS_NV_ALL_MEMCLKS
AGESA_IDS_NV_MEMORY_POWER_DOWN_MODE
IDSOPT_TRACING_ENABLED
IDS_HDT_CONSOLE
IDS_HDT_CONSOLE_INIT
IDS_HDT_CONSOLE_EXIT
CONSOLE
IDSOPT_TRACE_USER_OPTIONS
IDSOPT_PERF_ANALYSIS
IDS_HDT_CONSOLE
IDS_PERF_TIMESTAMP
IDS_PERF_ANALYSE
IDSOPT_HEAP_CHECKING
Heap_Check()
IDSOPT_IDT_EXCEPTION_TRAP
IDS_EXCEPTION_TRAP

Chapter 20 IDS Porting

The OptionsIds.h file is the file that controls which of the Integrated Debug Services will be included into the build. This file must be located in the build tip directory, the same location as the <plat>Options.c file. If the OptionsIds.h file does not exist in that directory, all IDS features will be disabled.

20.1 Check List

Points to remember:

- Create the OptionsIds.h file in the platform tip build directory. AMD recommends the use of a sub-directory named ‘AGESA’ to contain these files and the build output files.
- Edit and modify the option selections to meet the needs of the specific platform.
- Set the environment variable ‘AGESA_OptsDir’ to the path of the AGESA subdirectory of the platform tip build directory.
- Check to make sure the call-out “AgesaGetIdsData” on page 162 has been implemented in the host environment.
- Optionally, #define IDS_DEBUG_PORT to redirect AGESA_TESTPOINT output from the default IO port.

20.2 Example

Example OptionsIds.h file:

```
/**
 *
 * IDS Option File
 *
 * This file is used to switch on/off IDS features.
 *
 */
#ifndef _OPTION_IDS_H_
#define _OPTION_IDS_H_


#define IDSOPT_IDS_ENABLED          FALSE
#define IDSOPT_ERROR_TRAP_ENABLED   FALSE
#define IDSOPT_ASSERT_ENABLED       FALSE
#define IDSOPT_CONTROL_ENABLED      FALSE
#define IDSOPT_HEAP_CHECKING        FALSE
#define IDSOPT_TRACING_ENABLED      FALSE
#define IDSOPT_PERF_ANALYSIS        FALSE
```

```
#endif
```

You may cut and paste from this example to create the platform tip copy of the OptionsIds.h file.

20.3 Debugging Using IDS

It is highly recommended that you set the following for initial integration and development:

```
#define IDSOPT_IDS_ENABLED          TRUE
#define IDSOPT_ERROR_TRAP_ENABLED   TRUE
#define IDSOPT_ASSERT_ENABLED       TRUE
```

Enabling ASSERT will provide limit and sanity checking of many platform BIOS inputs, as well as AGESA™ internal error checks. Enabling IDS_ERROR_TRAP provides for debugging system error events. In some cases, these errors are really caused by incorrect wrapper implementation. Both macros provide for halting near to the point the error is actually detected and help to ensure issues don't go unnoticed.

When halting due to an ASSERT or IDS_ERROR_TRAP, the stop code will be displayed on the debug port. The stop code consists of two pieces of information: the file code and the line number. The file code is four hex digits and uniquely identifies the file containing the ASSERT or IDS_ERROR_TRAP. The file Include\Filecode.h contains the file code for each file. The line number is the remaining four hex digits and provides the line number in that file. For example, 0210 identifies line two hundred ten as the source line. Each ASSERT or IDS_ERROR_TRAP has adjacent comments in the source code describing the error and any wrapper issues known to be associated with the error.

The implementation provides a rotating display of the error code on the port 80h POST display card. The rotation is used to make it easier to view the error on both a 16-bit as well as a 32-bit display card. Additional support is available when running simulation.

In addition, AGESA_TESTPOINT output provides a useful progress indicator. Testpoints are defined in Ids.h, and are displayed on the IO port indicated by IDS_DEBUG_PORT. It is highly recommended that IDS_DEBUG_PORT be defined to a value that permits inspection, if the default port does not provide that. IDS_DEBUG_PORT only affects AGESA_TESTPOINT output, not other IDS debug output.

Chapter 21 HDT Console

This category of IDS Services allow the AMD Hardware Debug Tool (HDT) to interact with the AGESA™ software as it is performing its operation. Information tracing the progress of an algorithm can be displayed on the HDT monitor, providing data for analyzing the correctness and effectiveness of the algorithm or identifying platform hardware problem.

Refer to the file, AMDTools\HDTOUT\Readme.txt, in your release package for latest and most detailed information.

21.1 Starting an HDT Console debug session

The following components must be located and installed:

Hardware Requirements:

- HDT interface device. AMD provides several generations of devices for interfacing a host PC to the target test system. The presently in use devices are: Possum, Purple Possum or Wombat.
- USB cable (A <-> B)
- Motherboard of the test target must have the HDT connection header built onto the board
- Test target system
- Host PC

Software Requirements:

- HDT v7.5 or later
- Perl v5.10 or later (ActivePerl is recommended)
- BIOS with HDT Console feature enabled. To enable a BIOS with HDT Console features:
  - Ensure $(AGESA_OptsDir)\OptionsIds.h exists.
  - IDSOPT_IDS_ENABLED must be TRUE.
  - IDSOPT_TRACING_ENABLED must be TRUE.

On the host PC, open a command prompt window (‘DOS box’); set the path to the location of the desired script; then run the perl program with the desired script, e.g.

C:> perl hdtout2008.pl

When a BIOS with HDT Console features is loaded and executed, it will boot normally except that the HDT script can display the data on the console and also recorded to a text file on C:\ drive.

21.2 Available Scripts

AMD provides scripts in the \AMDTools sub-directory of the release package. These are the scripts available and the features they provide:

hdtout2008.pl - The main purpose for this script is to control, collect and display algorithm trace data.

perfout.pl - This script collects time stamp data from the test target and then analyses the data to produce some statistical time measurements.

Section VI - Appendices

Appendix A Tools

A.1 BINUTIL2

Command Line

```
binutil2 [options] /INFILE:[file] /OUTFILE:[file]
```

Description

This tool is used to transform the relocatable execution program file into a binary image. The process changes all of the relocatable address references into fixed address references.

Parameters

/INFILE

List of files to merge into the output image.

/OUTFILE

Name to file to create/replace which contains the merges modules.

[options]:

/IMAGEBASE:[Address] - Image Base address (Example /IMAGEBASE:0xffff0000)

/IMAGETYPE:[Type] - ImageType string B1/B2 (Example /IMAGEBASE:B1)

/VERSION:[Version] - Binary Image Version string (Example /VERSION:1.2.3)

/CREATORID:[Creatorid] - Creator ID string (Example /CREATORID:AMD)

/BDIFILE:[path/Filename] – Generate HDT symbol debug reference BDI file.

Usage Examples

```
$(AGESA_ROOT)AMDTools\binutil2 /INFILE:$(OutDir)Agesa_B2_Target.dll
   /OUTFILE:$(OutDir)agesa.b2 /IMAGETYPE:B2 /IMAGEBASE:$(AGESA_B2_ADDRESS)
   /BDIFILE:$(OutDir)Agesa_B2.bdi
```

Appendix B Logged Error Messages

Refer to the internal documentation for the latest and most detailed information (“Internal Documentation” on page 25) on specific logged events.

General form of numeric errors:

```
Class: AGESA_WARNING
SubClass: 0x00000000
0x00000000,       0x00000000,       0x00000000,       0x00000000  ...
      A                 B                 C                 D
```

Class—This matches the AGESA™ software return code reported from the entry procedure.

Sub-Class—This indicates the group of code files from which the error is being reported and the specific sub-class error. This value has the following bit fields:

AppId:8 — (MSB) Application that generated the event
EventId:8 — Nature of Event; dependent on the application
AppFunction:8 — Indicates the function where the event occurred
SubReason:8 — (LSB) Reason code for the Event; dependent on the application.

A, B, C, **D**—Sub-class data. This is data that better explains the error and/or its cause. The meaning of this data is specific to the sub-class. Please see the references below for details.

B.1 AGESA_SUCCESS Class

Sub-Class: Memory

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x04	0x01	0x26	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_EVENT_CAPSULE_IN_EFFECT					
0x04	0x02	0x26	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_EVENT_CONTEXT_RESTORE_IN_EFFECT					

Sub-Class: Processor

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x01	0x01	0x00	0x00000000	0x00000000	0x00000000	0x00000000
	CPU_ERROR_HEAP_IS_ALREADY_INITIALIZED - Heap has previously been initialized						

B.2 AGESA_BOUNDS_CHK Class

Sub-Class: Processor

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x02	0x01	0x00	0x00000000	0x00000000	0x00000000	0x00000000
	CPU_ERROR_HEAP_IS_FULL - Heap allocation for specified buffer handle failed as heap is full						
0x08	0x03	0x01	0x00	0x00000000	0x00000000	0x00000000	0x00000000
	CPU_ERROR_HEAP_BUFFER_HANDLE_IS_ALREADY_USED - Allocation incomplete as buffer has previously been allocated						
0x08	0x04	0x01	0x00	0x00000000	0x00000000	0x00000000	0x00000000
	CPU_ERROR_HEAP_BUFFER_HANDLE_IS_NOT_PRESENT - Unable to locate buffer handle or deallocate heap as buffer handle cannot be located						
0x08	0x00	0x01	0x00	0x00000000	0x00000000	0x00000000	0x00000000
	CPU_ERROR_HEAP_BUFFER_IS_NOT_PRESENT - Unable to locate pointer to the heap buffer						

B.3 AGESA_ALERT Class

Sub-Class: Memory

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x04	0x01	0x00	0x00	Socket	DCT	Channel	0x00000000
	MEM_ALERT_USER_TMG_MODE_OVERRULED The timing mode selected by the user via a BLDCFG_item could not be used for the present configuration and has been overruled.						
0x04	0x01	0x01	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_ORG_MISMATCH_DIMM						
0x04	0x01	0x02	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_BK_INT_DIS - Bank interleaving disable for internal issue						

Sub-Class: Processor

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x00	0x05	0x00	BIST Error	Socket	Core	0x00000000
	CPU_EVENT_BIST_ERROR - BIST error is reported on the specified socket and core.						

B.4 AGESA_WARNING Class

Sub-Class: Memory

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x04	0x01	0x16	0x00	Socket	DCT	Channel	0x00000000
	MEM_WARNING_UNSUPPORTED_QRDIMM - QR DIMMs detected but not supported						
0x04	0x02	0x16	0x00	Socket	DCT	Channel	0x00000000
	MEM_WARNING_UNSUPPORTED_UDIMM - Unbuffered DIMMs detected but not supported						
0x04	0x03	0x16	0x01	Socket	DCT	Channel	0x00000000
	MEM_WARNING_UNSUPPORTED_SODIMM - SO-DIMMs detected but not supported						

0x04	0x04	0x16	0x00	Socket	DCT	Channel	0x00000000
		MEM_WARNING_UNSUPPORTED_X4DIMM - x4 DIMMs detected but not supported					
0x04	0x04	0x17	0x02	Socket	DCT	Channel	0x00000000
		MEM_ERROR_ECC_DIS					
0x04	0x05	0x16	0x00	Socket	DCT	Channel	0x00000000
		MEM_WARNING_UNSUPPORTED_RDIMM - Registered DIMMs detected but not supported					
0x04	0x06	0x16	0x00	Socket	DCT	Channel	0x00000000
		MEM_WARNING_UNSUPPORTED_LRDIMM - Load Reduced DIMMs detected but not supported					
0x04	0x01	0x19	0x00	Socket	DCT	Channel	DimmId
		MEM_WARNING_NO_SPDTRC_FOUND					
0x04	0x01	0x17	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_EMP_NOT_SUPPORTED - Processor is not capable for EMP					
0x04	0x02	0x17	0x01	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_EMP_CONFLICT - EMP cannot be enabled if channel interleaving, bank interleaving, or bank swizzle is enabled.					
0x04	0x03	0x17	0x02	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_EMP_NOT_ENABLED - Memory size is not power of two.					
0x04	0x01	0x18	0x00	Socket	DCT	Channel	0
		MEM_WARNING_PERFORMANCE_ENABLED_BATTERY_LIFE_PREFERRED					
0x04	0x02	0x19	0x00	Socket	DCT	Channel	DIMM
		MEM_WARNING_SPD_MODDLY_OUT_OF_RANGE					
0x04	0x01	0x20	0x00	Socket	DCT	Channel	0
		MEM_WARNING_NODE_INTERLEAVING_NOT_ENABLED					
0x04	0x01	0x21	0x00	Socket	DCT	Channel	0
		MEM_WARNING_CHANNEL_INTERLEAVING_NOT_ENABLED					
0x04	0x01	0x22	0x00	Socket	DCT	Channel	0
		MEM_WARNING_BANK_INTERLEAVING_NOT_ENABLED					
0x04	0x01	0x23	0x00	Socket	DCT	Channel	0
		MEM_WARNING_VOLTAGE_1_35_NOT_SUPPORTED					
0x04	0x01	0x24	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_INITIAL_DDR3VOLT_NONZERO					
0x04	0x01	0x25	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_NO_COMMONLY_SUPPORTED_VDDIO					
0x04	0x01	0x26	0x00	Socket	0x00000000	0x00000000	0x00000000

		MEM_WARNING_MBIST_REQUIRED_FUNCTION_NOT_SUPPORTED					
0x04	0x01	0x27	0x00	Socket	0x00000000	0x00000000	0x00000000
		MEM_WARNING_MBIST_INTERNAL_ERROR					
0x04	0x01	0x28	0x00	Socket	Channel	Chipselect	Subtest
		MEM_WARNING_MBIST_DEFAULT_TEST_FAIL					
0x04	0x01	0x29	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_AMP_SUPPORT_DETECTED_BUT_NOT_ENABLED					
0x04	0x02	0x29	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_WARNING_AMP_SELECTED_BUT_NOT_ENABLED					

Sub-Class: Processor

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x00	0x21	0xRR	Data A	Data B	Data C	0x00000000
		CPU_EVENT_EXECUTION_CACHE_ALLOCATION_ERROR - Execution cache allocation warning, Execution cache allocation error. RR — Allocation rule number that has been violated 0x01 AGESA_CACHE_SIZE_REDUCED 0x02 AGESA_CACHE_REGIONS_ACROSS_1MB 0x03 AGESA_CACHE_REGIONS_ACROSS_4GB Data A — cache region index in parameter set that violated the rule Data B — cache region start address Data C — cache region size					
0x08	0x00	0x22	0x00	Requested	Actual Level	0x00000000	0x00000000
		CPU_WARNING_ADJUSTED_LEVELING_MODE					
0x08	0x00	0x04	0x00	Socket	Pstate	0x00000000	0x00000000
		CPU_EVENT_PM_PSTATE_OVERCURRENT					
0x08	0x05	0x01	0x00	Socket	Raw CPUID	0x00000000	0x00000000
		CPU_EVENT_UNKNOWN_PROCESSOR_REVISION					

Sub-Class: AMD_ENABLE_STACK

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x02	0x05	0x00	--	--	--	--
	CPU_EVENT_STACK_REENTRY - The stack has already been enabled and this is a redundant invocation of AMD_ENABLE_STACK. There is no event logged and no data values. The event sub-class is returned along with the status code.						

B.5 AGESA_ERROR Class

Sub-Class: Memory

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x04	0x01	0x03	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_NO_DQS_POS_RD_WINDOW — No DQS Position window for RD DQS						
0x04	0x02	0x03	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_SMALL_DQS_POS_RD_WINDOW— Small DQS Position window for RD DQS						
0x04	0x03	0x03	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_NO_DQS_POS_WR_WINDOW						
0x04	0x04	0x03	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_SMALL_DQS_POS_WR_WINDOW						
0x04	0x01	0x05	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_DIMM_SPARING_NOT_ENABLED — DIMM sparing has not been enabled for an internal issues						
0x04	0x05	0x03	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_RCVR_EN_VALUE_TOO_LARGE — Receive Enable value is too large						
0x04	0x06	0x03	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_RCVR_EN_NO_PASSING_WINDOW						
0x04	0x01	0x06	0x00	Socket	DCT	Channel	0x00000000
	MEM_ERROR_DRAM_ENABLED_TIME_OUT						

0x04	0x01	0x07	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_DCT_ACCESS_DONE_TIME_OUT					
0x04	0x01	0x08	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_SEND_CTRL_WORD_TIME_OUT					
0x04	0x01	0x09	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_PREF_DRAM_TRAIN_MODE_TIME_OUT					
0x04	0x01	0x0A	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_ENTER_SELF_REF_TIME_OUT					
0x04	0x01	0x0B	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_FREQ_CHG_IN_PROG_TIME_OUT					
0x04	0x02	0x0A	0x00	Socket	DCT	Channel	
		MEM_ERROR_EXIT_SELF_REF_TIME_OUT					
0x04	0x01	0x0C	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_SEND_MRS_CMD_TIME_OUT					
0x04	0x01	0x0D	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_SEND_ZQ_CMD_TIME_OUT					
0x04	0x01	0x0E	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_DCT_EXTRA_ACCESS_DONE_TIME_OUT					
0x04	0x01	0x0F	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_MEM_CLR_BUSY_TIME_OUT					
0x04	0x02	0x0F	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_MEM_CLEARED_TIME_OUT					
0x04	0x01	0x10	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_FLUSH_WR_TIME_OUT					
0x04	0x07	0x03	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_MAX_LAT_NO_WINDOW					
0x04	0x08	0x03	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_PARALLEL_TRAINING_LAUNCH_FAIL					
0x04	0x09	0x03	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_PARALLEL_TRAINING_TIME_OUT					
0x04	0x01	0x11	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_NO_ADDRESS_MAPPING					
0x04	0x0A	0x03	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_RCVR_EN_NO_PASSING_WINDOW_EQUAL_LIMIT					
0x04	0x0B	0x03	0x00	Socket	DCT	Channel	0x00000000

		MEM_ERROR_RCVR_EN_VALUE_TOO_LARGE_LIMIT_LESS_ONE					
0x04	0x01	0x12	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_CHECKSUM_NV_SPDCHK_RESTRT_ERROR					
0x04	0x01	0x13	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_NO_CHIPSELECT					
0x04	0x01	0x15	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_UNSUPPORTED_333MHZ_UDIMM					
0x04	0x02	0x25	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_VDDIO_UNSUPPORTED					
0x04	0x0D	0x03	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_NO_2D_RDDQS_WINDOW					
0x04	0x0E	0x03	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_NO_2D_RDDQS_HEIGHT					
0x04	0x0F	0x03	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_2D_BQS_ERROR					
0x04	0x02	0x24	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_INVALD_2D_RDDQS_VALUE					
0x04	0x02	0x23	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_2D_DQS_VREF_MARGIN_ERROR					
0x04	0x01	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_LR_IBT_NOT_FOUND					
0x04	0x02	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_MR0_NOT_FOUND					
0x04	0x03	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_ODT_PATTERN_NOT_FOUND					
0x04	0x04	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_RC2_IBT_NOT_FOUND					
0x04	0x05	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_RC10_OP_SPEED_NOT_FOUND					
0x04	0x06	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_RTT_NOT_FOUND					
0x04	0x07	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_P2D_NOT_FOUND					
0x04	0x08	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_SAO_NOT_FOUND					

0x04	0x09	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_CLK_DIS_MAP_NOT_FOUND					
0x04	0x0A	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_CKE_TRI_MAP_NOT_FOUND					
0x04	0x0B	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_ODT_TRI_MAP_NOT_FOUND					
0x04	0x0C	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_CS_TRI_MAP_NOT_FOUND					
0x04	0x0D	0x35	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_TRAINING_SEED_NOT_FOUND					
0x04	0x0D	0x1F	0x00	Socket	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_CRAT_MEM_AFFINITY					
0x04	0x0E	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_DEALLOCATE_FOR_DATAEYE					
0x04	0x0F	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_DATAEYE					
0x04	0x0E	0x35	0x00	Socket	Channel	Chipselect	Subtest
		MEM_ERROR_MBIST_TEST_FAIL					
0x04	0x01	0x26	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_NBPSTATE_TRANSACTION_TIME_OUT Incorrect Phy master channel setting.					
0x04	0x0D	0x04	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_NO_2D_WRDAT_WINDOW					
0x04	0x0E	0x04	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_NO_2D_WRDAT_HEIGHT					
0x04	0x0F	0x04	0x00	Node	DCT	Channel	0x00000000
		MEM_ERROR_2D_WRDAT_ERROR					
0x04	0x10	0x04	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_INVALID_2D_WRDAT_VALUE					
0x04	0x11	0x04	0x00	Node	DCT	Channel	0x00000000
		MEM_ERROR_2D_WRDAT_VREF_MARGIN_ERROR					
0x04	0x12	0x04	0x00	Node	DCT	Channel	0x00000000
		MEM_ERROR_PMU_TRAINING					
0x04	0x10	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000

		MEM_ERROR_HEAP_ALLOCATE_FOR_PMU_SRAM_MSG_BLOCK					
0x04	0x11	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_DEALLOCATE_FOR_PMU_SRAM_MSG_BLOCK					
0x04	0x12	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_LOCATE_FOR_PMU_SRAM_MSG_BLOCK					

Sub-Class: Processor

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x00	0x02	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		CPU_ERROR_BRANDID_HEAP_NOT_AVAILABLE - No heap is allocated for BrandId structure					
0x08	0x00	0x03	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		CPU_ERROR_MICRO_CODE_PATCH_IS_NOT_LOADED- Unable to load micro code patch					
0x08	0x02	0x04	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		CPU_ERROR_PSTATE_HEAP_NOT_AVAILABLE - No heap is allocated for the Pstate structure					
0x08	0x03	0x04	0x00	Socket	Index	0x00000000	0x00000000
		CPU_ERROR_PM_NB_PSTATE_MISMATCH- NB P-state indicated by Index was disabled due to mismatch between processors.					
0x08	0x00	0x21	0xRR	Data A	Data B	Data C	0x00000000
		CPU_EVENT_EXECUTION_CACHE_ALLOCATION_ERROR - Execution cache allocation warning, Execution cache allocation error. RR — Allocation rule number that has been violated 0x04 AGESA_REGION_NOT_ALIGNED_ON_BOUNDARY 0x05 AGESA_START_ADDRESS_LESS_D0000 0x06 AGESA_THREE_CACHE_REGIONS_ABOVE_1MB 0x07 AGESA_DEALLOCATE_CACHE_REGIONS Data A — cache region index in parameter set that violated the rule Data B — cache region start address Data C — cache region size					
0x08	0x00	0x24	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		CPU_EVENT_SCS_INITIALIZATION_ERROR					
0x08	0x00	0x24	0x01	0x00000000	0x00000000	0x00000000	0x00000000

		CPU_EVENT_SCS_INITIALIZATION_ERROR - Heap entry missing					
0x08	0x00	0x24	0x02	0x00000000	0x00000000	0x00000000	0x00000000
		CPU_EVENT_SCS_INITIALIZATION_ERROR - Buffer Empty					
0x08	0x00	0x24	0x03	0x00000000	0x00000000	0x00000000	0x00000000
		CPU_EVENT_SCS_INITIALIZATION_ERROR - Weights Mismatch					
0x08	0x04	0x04	0x00	Input Limit	PState Freq original/new	0x00000000	0x00000000
		CPU_ERROR_PM_ALL_PSTATE_OVER_FREQUENCY_LIMIT - all P-states are over the input frequency limit. The minimum P-state frequency will be used.					
0x08	0x05	0x4	0x00	Input Limit	PState Freq original/new	Boost Pstate# original/new	SW Pstate original/new
		CPU_EVENT_PM_PSTATE_FREQUENCY_LIMIT - successful					

Sub-Class: GNB

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x02	0x01	0x00	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_INVALID_CONFIGURATION — Invalid user input for configuration.					
0x02	0x01	0x00	0x01	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_INVALID_PCIE_TOPOLOGY_CONFIGURATION — Invalid user input for configuration.					
0x02	0x01	0x00	0x02	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_INVALID_PCIE_PORT_CONFIGURATION — Invalid user input for configuration.					
0x02	0x01	0x00	0x03	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_INVALID_DDI_LINK_CONFIGURATION — Invalid user input for configuration.					
0x02	0x01	0x00	0x04	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_INVALID_LINK_WIDTH_CONFIGURATION — Invalid user input for configuration.					
0x02	0x01	0x00	0x05	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_INVALID_LANES_CONFIGURATION — Invalid user input for configuration.					
0x02	0x01	0x00	0x06	0x00000000	0x00000000	0x00000000	0x00000000

		GNB_EVENT_INVALID_DDI_TOPOLOGY_CONFIGURATION — Invalid user input for configuration.					
0x02	0x02	0x00	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_LINK_TRAINING_FAIL — PCIe® Link training failure.					
0x02	0x03	0x00	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_BROKEN_LANE_RECOVERY — Broken lane workaround applied.					
0x02	0x04	0x00	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		GNB_EVENT_GEN2_SUPPORT_RECOVERY — dropped to Gen 1.					

B.6 AGESA_CRITICAL Class

Sub-Class: Memory

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x04	0x09	0x1F	0x00	Socket	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_DMI_TABLE_DDR3					
0x04	0x0A	0x1F	0x00	Socket	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_DMI_TABLE_DDR2					
0x04	0x01	0x14	0x00	Socket	0x00000000	0x00000000	0x00000000
		MEM_ERROR_UNSUPPORTED_DIMM_CONFIG					

B.7 AGESA_FATAL Class

Sub-Class: Memory

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x04	0x01	0x1A	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_MINIMUM_MODE — Running in minimum mode.					
0x04	0x01	0x1B	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_MODULE_TYPE_MISMATCH_DIMM — DIMM modules are mismatched.					

0x04	0x01	0x1C	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_NO_DIMM_FOUND_ON_SYSTEM — No DIMMs have been found.					
0x04	0x01	0x1D	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_MISMATCH_DIMM_CLOCKS — DIMM clocks miss-matched.					
0x04	0x01	0x1E	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_NO_CYC_TIME — No cycle time found.					
0x04	0x01	0x1F	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_DYN_STORING_OF_TRAINED_TIMINGS					
0x04	0x02	0x1F	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_DCT_STRUCT_AND_CH_DEF_STRUCT					
0x04	0x03	0x1F	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_REMOTE_TRAINING_ENV					
0x04	0x04	0x1F	0x00	Socket	DCT	Channel	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_SPD					
0x04	0x05	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_RECEIVED_DATA					
0x04	0x06	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_S3_SPECIAL_CASE_REGISTERS					
0x04	0x07	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_TRAINING_DATA					
0x04	0x02	0x23	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_IDENTIFY_DIMM_MEM_NB_BLOCK					
0x04	0x0B	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_ALLOCATE_FOR_2D					
0x04	0x0C	0x1F	0x00	0x00000000	0x00000000	0x00000000	0x00000000
		MEM_ERROR_HEAP_DEALLOCATE_FOR_2D					

Sub-Class: Processor

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x01	0x04	0x00	Socket	0x00000000	0x00000000	0x00000000
		CPU_EVENT_PM_ALL_PSTATES_OVERCURRENT - All PStates exceeded the motherboard current limit on specified socket					

Sub-Class: AMD_ENABLE_STACK

Sub-Class Fields				Sub-Class Data Values			
ID	**Event**	**Fcn**	**Sub**	**A**	**B**	**C**	**D**
0x08	0x01	0x05	0x00	--	--	--	--
	CPU_EVENT_UNKNOWN_PROCESSOR_FAMILY - Stack cannot be enabled due to unknown processor family. There is no event logged and no data values. The sub-class is returned along with the status code.						
0x08	0x03	0x05	0x00	--	--	--	--
	CPU_EVENT_CORE_NOT_IDENTIFIED - Stack cannot be enabled due to core numbering issue, such being unable to determine compute unit primary status. There is no event logged and no data values. The sub-class is returned along with the status code.						

Appendix C Memory Details

Refer to the internal documentation for the latest and most detailed information (“Internal Documentation” on page 25).

C.1 Modifying or Correcting SPD values

The memory code uses the initializer function to load the SPD data for all of the memory DIMMs in the system prior to the entry for the main memory configuration process. Once the initializer completes, control is returned to the host environment. At this point in time, modifications or corrections can be made to the SPD data that will be used by the main configuration process. Be careful to maintain a valid checksum for the DIMM data.

C.2 Specialized Platform Files

The AGESA™ software attempts to automatically identify timing parameters. However, variations in the motherboard design are unpredictable. The AGESA™ software includes small procedures that contain platform-specific settings that **may** need to be modified or replaced for certain motherboards. The AMD Reference Design Kit (ARDK) motherboards follow the AMD platform design guidelines and the default platform description procedures included in the AGESA™ software use timing values that match these design guidelines. Motherboards that do not conform to the AMD design guidelines may need to supply replacement procedures.

The AGESA™ memory software segregates procedures for memory timing by each type of memory (Registered or Unbuffered) and technology (DDR2 or DDR3). These are accessed through an array of procedure pointers contained in a MEM_DATA_STRUCT structure element:

```
AGESA_STATUS (*GetPlaformCfg[]) (
                  MEM_DATA_STRUCT   *MemData,
                  UINT8             SocketID,
                  CH_DEF_STRUCT     *CurrentChannel;
```

The host environment must replace the default procedure pointers with pointers to their private procedures that conform to the following definition.

MemAGetPsCfgRHy2
MemAGetPsCfgRHy3

Internal routines to obtain platform-specific address and command settings for DIMM configurations.

Prototype

```
AGESA_STATUS
MemAGetPsCfgRHy3 (
     IN OUT  MEM_DATA_STRUCT   *MemData,
     IN      UINT8             SocketID,
     IN OUT  CH_DEF_STRUCT     *CurrentChannel);
);
```

Parameters

MemData

Pointer to a data structure containing the memory information.

SocketId

The socket number for the memory presently being configured.

CurrentChannel

The memory channel number for the memory presently being configured.

Related Definitions

```
MEM_DATA_STRUCT
CH_DEF_STRUCT
```

These are the same data structures used by the main memory routine. See “AmdInitPost” on page 114 for structure details.

Description

This procedure is called from the AGESA™ software main code to determine platform-specific memory timing values. The host environment may choose to replace this function with their own version to adjust timing parameters specific to the board. This procedure, or its replacement, has the entire content of the MEM_DATA_STRUCT available for reference input. At a minimum, the procedure needs to set or adjust the following timing elements in that structure:

DCTData->**SlowMode** Indicates if Slow Access mode is supported (also known as 1T or 2T timing mode)
DCTData->**OdcCtl** Output driver control value or “drive strength”
DCTData->**AddrTmg** Address/data timing values

Dependencies

The platform must conform to the AMD Platform Design Guide and approximate the AMD reference design board for the default functions to perform properly. If the platform deviates from the reference platform characteristics, then the host environment must tailor this function to the specific needs of the platform and replace the function pointer in the parameter block before making the call to the main memory function.

Status Codes Returned

AGESA_SUCCESS	The function has completed successfully.
AGESA_ERROR	

C.3 Advanced DQS Training

Note:“BLDCFG_DQS_TRAINING_CONTROL” on page 209— False means that the platform has stored values previously and intends to use them. If the stored values are not present (or not valid) then active training will be used.

C.4 On-Line Spares

This is a mechanism in which a DIMM of memory is reserved for use as a “spare.” When ECC errors reach a threshold level in a single DIMM, the active DIMM is swapped out of use and the “spare” DIMM is swapped into use in its place. This swap is done by the hardware and is near instantaneous. Further details can be found in the BIOS and Kernel Developers Guide (BKDG).

To activate the feature, refer to the build option “BLDOPT_REMOVE_ONLINE_SPARE_SUPPORT” on page 178, and the build configuration element “BLDCFG_ONLINE_SPARE” on page 207.

C.5 Platform Specific Override

The Platform Specific Override Interface provides additional, expert level, customizations to the AGESA™ Memory code.

The Platform Specific Override customizations are made using the default table located in the platform options C file. Each platform options file contains a default override table called DefaultPlatformMemoryConfiguration. This table may be modified or extended to customize for your platform. The table may also be replaced by updating the parameter with the pointer to the replacement table when calling AmdInitPost, see “PlatformMemoryConfiguration” on page 118.

Customizations are made in the table using a robust set of macros. These are presented below in three groups: basic macros, conditional macros, and table macros. Any memory settings which are not customized using the macros will be initialized using recommended settings.

C.5.1 Expert Overrides

The following macros provide expert platform specific customizations.

MOTHER_BOARD_LAYERS

Prototype

```
MOTHER_BOARD_LAYERS (Layers)
```

Parameters

Layers

Customize the number of board layers for the system.

LAYERS_4 - The system has 4 board layers (default).
LAYERS_6 - The system has 6 board layers.

Description

This override allows correct customization of memory settings which vary based on the number of board layers. Address, Command, and Drive Strength memory settings may be affected.

MEMCLK_DIS_MAP

Prototype

```
  MEMCLK_DIS_MAP (SocketID, ChannelID, Bit0Map, Bit1Map, Bit2Map, Bit3Map,
Bit4Map, Bit5Map, Bit6Map, Bit7Map)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

Bit[7:0]Map

Indicating the memory clock pins connect to which memory chip-selects / ranks.

Description

The memory clock pins are identified based on BKDG definition of MEM_CLK_DIS bitmap. Disabling unused MemClk may save power.

For example, BKDG definition of MEM_CLK_DIS bitmap is like below:

```
Bit   pin name
0    M[B,A]_CLK_H/L[0]
1    M[B,A]_CLK_H/L[1]
2    M[B,A]_CLK_H/L[2]
3    M[B,A]_CLK_H/L[3]
4    M[B,A]_CLK_H/L[4]
5    M[B,A]_CLK_H/L[5]
6    M[B,A]_CLK_H/L[6]
7    M[B,A]_CLK_H/L[7]
```

And platform has the following routing:

```
CS0   M[B,A]_CLK_H/L[4]
CS1   M[B,A]_CLK_H/L[2]
CS2   M[B,A]_CLK_H/L[3]
CS3   M[B,A]_CLK_H/L[5]
```

Then platform can specify the following macro:

```
MEMCLK_DIS_MAP(ANY_SOCKET, ANY_CHANNEL, 0x00, 0x00, 0x02, 0x04, 0x01, 0x08,
0x00, 0x00)
```

ON_DIMM_THERMAL_CONTROL

Prototype

```
ON_DIMM_THERMAL_CONTROL (SocketID, ChannelID, EnableDisable)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

EnableDisable

Indicate whether to enable on DIMM thermal sensor support.

ENABLE_ - Enable the feature on DIMMs for the channel specified.
DISABLE_ - Disable the feature on DIMMs for the channel specified.

Description

Support for on DIMM thermal sensors can be overridden.

CKE_TRI_MAP

Prototype

```
CKE_TRI_MAP (SocketID, ChannelID, Bit0Map, Bit1Map, Bit2Map, Bit3Map)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

Bit[3:0]Map

Indicating CKE signals connect to which memory chip-selects/ranks.

Description

The CKE pins are identified based on BKDG definition of CKETri bitmap. Tri-state of unused CKE may save power.

ODT_TRI_MAP

Prototype

```
ODT_TRI_MAP (SocketID, ChannelID, Bit0Map, Bit1Map, Bit2Map, Bit3Map)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

Bit[3:0]Map

Indicating ODT signals connect to which memory chip-selects/ranks.

Description

The ODT pins are identified based on BKDG definition of ODTTri bitmap. Tri-state of unused ODT pins may save power.

CS_TRI_MAP

Prototype

```
CS_TRI_MAP (SocketID, ChannelID, Bit0Map, Bit1Map, Bit2Map, Bit3Map, Bit4Map,
Bit5Map, Bit6Map, Bit7Map)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

Bit[7:0]Map

Indicating chip selects signals connect to which memory chip-selects/ranks.

Description

The chip select pins are identified based on BKDG definition of ChipSelTri bitmap. Tri-state of unused chip selects signals may save power.

NUMBER_OF_DIMMS_SUPPORTED

Prototype

```
NUMBER_OF_DIMMS_SUPPORTED (SocketID, ChannelID, NumberOfDimmSlotsPerChannel)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

NumberOfDimmSlotsPerChannel

Specifies the number of DIMM slots per channel.

Description

This macro usually exists in the platform options file to specify the number of DIMM slots per channel in socket/channel basis. On a platform with soldered down DRAM only, SODIMM only, or soldered-down DRAM plus SODIMM, this macro will be used to specify total number of DIMMs supported on a channel, including slotted and soldered down DIMMs.

NUMBER_OF_SOLDERED_DOWN_DIMMS_SUPPORTED

Prototype

```
  NUMBER_OF_SOLDERED_DOWN_DIMMS_SUPPORTED (SocketID, ChannelID,
NumberOfSolderedDownDimmsPerChannel)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

NumberOfSolderedDownDimmsPerChannel

Specifies the number of soldered down DIMMs per channel.

Description

This macro is used to specify the number of soldered-down DIMMs per channel in socket/ channel basis. This macro is similar to SOLDERED_DOWN_SODIMM_TYPE macro, however, it explicitly tells AGESA how many soldered-down DIMMs are on the board. It is recommended that this macro be used instead of SOLDERED_DOWN_SODIMM_TYPE macro when the board design is soldered-down DRAM(s) only or solder-down DRAM(s) plus SODIMM(s) configuration.

NUMBER_OF_CHIP_SELECTS_SUPPORTED

Prototype

```
NUMBER_OF_CHIP_SELECTS_SUPPORTED (SocketID, ChannelID,
NumberOfChipSelectsPerChannel)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

NumberOfChipSelectsPerChannel

Specifies the number of chip selects per channel.

Description

This macro is used to specify the number of chip selects per channel in socket/channel basis.

NUMBER_OF_CHANNELS_SUPPORTED

Prototype

```
NUMBER_OF_CHANNELS_SUPPORTED (SocketID, NumberOfChannelsPerSocket)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

NumberOfChannelsPerSocket

Specifies the number of channels per socket.

Description

This macro is used to specify the numbers of channels on socket SocketID.

OVERRIDE_DDR_BUS_SPEED

Prototype

```
OVERRIDE_DDR_BUS_SPEED (SocketID, ChannelID, TimingMode, BusSpeed)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

TimingMode

Indicating which timing mode that applies the desired bus speed. Possible values are: TIMING_MODE_LIMITED, TIMING_MODE_SPECIFIC

BusSpeed

Frequencies to apply change. Possible values are:

DDR667_FREQUENCY	DDR1866_FREQUENCY
DDR800_FREQUENCY	DDR2100_FREQUENCY
DDR1066_FREQUENCY	DDR2133_FREQUENCY
DDR1333_FREQUENCY	DDR2400_FREQUENCY
DDR1600_FREQUENCY	

Description

This macro is used to limit/force the memory speed of channel ChannelID on socket SocketID.

DRAM_TECHNOLOGY

Prototype

```
DRAM_TECHNOLOGY (SocketID, MemTechType)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

MemTechType

Indicating the DRAM technology. Possible values are :
DDR3_TECHNOLOGY, DDR2_TECHNOLOGY

Description

This macro specifies DRAM technology type of socket SocketID.

WRITE_LEVELING_SEED

Prototype

```
  WRITE_LEVELING_SEED (SocketID, ChannelID, DimmID, Byte0Seed, Byte1Seed,
Byte2Seed, Byte3Seed, Byte4Seed, Byte5Seed, Byte6Seed, Byte7Seed, ByteEccSeed)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

DimmIID

A mask indicates the physical channel. Possible values are:

DIMM0	DIMM3
DIMM1	ALL_DIMMS
DIMM2	

These values may be combined as follows: DIMM0 + DIMM1

Byte[7:0]Seed/ByteEccSeed

8 bits seed values for each byte lanes.

Description

This macro specifies the write leveling seed values for a specific DIMM on a channel of a socket. The default behavior when this macro is not populated: Defined by BKDG.

HW_RXEN_SEED

Prototype

```
  HW_RXEN_SEED (SocketID, ChannelID, DimmID, Byte0Seed, Byte1Seed, Byte2Seed,
Byte3Seed, Byte4Seed, Byte5Seed, Byte6Seed, Byte7Seed, ByteEccSeed)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

DimmIID

A mask indicates the physical channel. Possible values are:

DIMM0	DIMM3
DIMM1	ALL_DIMMS
DIMM2	

These values may be combined as follows: DIMM0 + DIMM1

Byte[7:0]Seed/ByteEccSeed

8 bits seed values for each byte lanes.

Description

This macro specifies the HW RXEN training seed values for a specific DIMM on a channel of a socket. The default behavior when this macro is not populated: Defined by BKDG.

NO_LRDIMM_CS67_ROUTING

Prototype

```
NO_LRDIMM_CS67_ROUTING (SocketID, ChannelID)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

Description

When this macro is stated, it tells AGESA that there is no CS6 and CS7 routing for LRDIMM

The default behavior when this macro is not populated: AGESA supports CS6 and CS7 routing for LRDIMM.

SOLDERED_DOWN_SODIMM_TYPE

Prototype

```
SOLDERED_DOWN_SODIMM_TYPE (SocketID, ChannelID)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

Description

When this macro is stated, it tells AGESA that current platform is soldered down SO-DIMM configuration. The default behavior when this macro is not populated: AGESA assumes that no soldered down SO-DIMMs supported.

MIN_RD_WR_DATAEYE_WIDTH

Prototype

```
   MIN_RD_WR_DATAEYE_WIDTH (SocketID, ChannelID, MinRdDataeyeWidth,
MinWrDataeyeWidth)
```

Parameters

SocketID

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelID

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

MinRdDataeyeWidth

A 8 bits value for minimum read data eye width.

MinWrDataeyeWidth

A 8 bits value for minimum write data eye width.

Description

This macro specifies minimum read and write date eye width for a channel of a socket. The default behavior when this macro is not populated: AGESA minimum width for DQS RD/QR training is 3 delay steps wide.

CPU_FAMILY_TO_OVERRIDE

Prototype

```
CPU_FAMILY_TO_OVERRIDE (CpuFamilyRevision)
```

Parameters

CpuFamilyRevision

A 32 bit CPUID value that indicates the CPU family and revision.

Description

This macro is a special macro, it allows customers stating CPU family specific platform memory overriding macros.

The platform memory overriding macros follow this macro will be treated as CPU family specific macros. For those platform memory overriding macros not follow this macro will be treated as global macros, that means, they are applied to all CPU families except for the CPU family that CPU_FAMILY_TO_OVERRIDE macro specifies.

Example:

```
// Global PSO for the CPU families which no exclusive PSO stated below
ODT_TRI_MAP(ANY_SOCKET, ANY_CHANNEL, 0x11, 0x22, 0x33, 0x44),

// Exclusive PSO macro for CPU family (0x00600F10)
CPU_FAMILY_TO_OVERRIDE (0x00600F10),
ODT_TRI_MAP(ANY_SOCKET, ANY_CHANNEL, 0x99, 0x88, 0x77, 0x66),

// Exclusive PSO macro for CPU family (0x00500F20)
CPU_FAMILY_TO_OVERRIDE (0x00500F20),
ODT_TRI_MAP(ANY_SOCKET, ANY_CHANNEL, 0x68, 0x79, 0x8A, 0x9B),
```

Dependencies

Not all of platform memory override macros are CPU family dependent, NUMBER_OF_DIMMS_SUPPORTED macro, for example, it is related to boards design.

Currently, the available platform memory override macros can be supported in CPU family/ revision basis are :

MEMCLK_DIS_MAP, CKE_TRI_MAP,
ODT_TRI_MAP, CS_TRI_MAP,
OVERRIDE_DDR_BUS_SPEED, WRITE_LEVELING_SEED,
HW_RXEN_SEED, MIN_RD_WR_DATAEYE_WIDTH

C.5.2 Conditional Overrides

The following macros provide conditional expert platform specific customizations, primarily for processor families for which table driven overrides are not available.

The following parameters can currently be overridden by this interface:

- "Dram ODT
- "Memory Address Timing
- "Dram Drive Strength
- "Slew Rate
- "DDR Frequency as a function of DIMM voltage

These overrides may be made conditional on the platform characteristics below, by preceding the configuration action macro with conditionals macros. A configuration action macro closes all conditionals preceding it.

- "Processor Socket
- "Memory Channel
- "Dimm Number
- "Frequency
- "Dimm population
- "Spd Contents

The macros below are divided into macros for specifying the conditions and macros for indicating the configuration actions.

C.5.2.1 Condition and Test Macros

CONDITION_AND

Prototype

```
CONDITION_AND
```

Parameters

<none>

Description

This macro opens a condition block. The conditions that follow will be logically ANDed together when evaluating whether to apply the associated actions. Multiple CONDITION_AND macros can be used in succession for a particular group of ACTION items. In this case, there is an implied OR of the separate CONDITION_AND blocks.

COND_LOC

Prototype

```
COND_LOC (SocketMsk, ChannelMsk, DimmMsk)
```

Parameters

SocketMsk

Mask indicating the physical processor socket or sockets. Possible values are:

SOCKET0	SOCKET5
SOCKET1	SOCKET6
SOCKET2	SOCKET7
SOCKET3	ANY_SOCKET
SOCKET4	

These values may be combined as follows: SOCKET0 + SOCKET2

ChannelMsk

A mask indicates the physical channel. Possible values are:

CHA	CHD
CHB	ANY_CHANNEL
CHC	

These values may be combined as follows: CHA + CHC

DimmMsk

A mask of the DIMMS to with the condition applies. Possible values are:

DIMM0	DIMM3
DIMM1	ALL_DIMM
DIMM2	

These values may be combined as follows: DIMM0 + DIMM3

Description

This specifies the Socket/Channel/Dimm to apply the changes. There can only be one COND_LOC block in each CONDITION_AND block and it must be the first entry in the block. The action entries defined in this block will be ignored unless the criteria in this field is met. However, certain actions may not be performed at the channel or dimm scope, and therefore will not be filtered on this basis. For instance, Address timing is programmed per channel and therefore will be applied as long as the socket and channel match. The Dimm field will be ignored.

COND_SPD

Prototype

```
COND_SPD (Byte, Mask, Value)
```

Parameters

Byte

The byte index of the spd value

Mask

The bit mask of the specific spd field

Value

The SPD value to match in this field.

Description

This entry tests a specific SPD value. As many of these as are desired can be grouped together in a CONDITION_AND block.

C.5.2.2 Conditional Action Macros

ACTION_ODT

Prototype

```
ACTION_ODT (Frequency, Dimms, QrDimms, DramOdt, QR_DramOdt, DramDynOdt)
```

Parameters

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

Dimms

Number of DIMMs in the channel. Possible values are:

NO_DIMM	THREE_DIMM
ONE_DIMM	FOUR_DIMM
TWO_DIMM	ANY_NUM

These values can be combined as follows: ONE_DIMM + TWO_DIMM

QrDimms

Number of quad rank DIMMs in the channel. Possible values are same as above.

DramOdt

Dram ODT (Rtt_Nom) as specified in the JEDEC DDR3 specification to be applied.

Qr_DramOdt

Dram ODT (Rtt_Nom) as specified in the JEDEC DDR3 specification to be applied to quad rank dims

DramDynOdt

Dram Write ODT (Rtt_Wr) as specified in the JEDEC DDR3 specification to be applied

ACTION_ADDRTMG

Prototype

```
ACTION_ADDRTMG (Frequency, DimmConfig, AddrTmg)
```

Parameters

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

DimmConfig

Mask indicating DIMM configuration. Possible values are:

ANY_	DR_DIMM1	QR_DIMM3
SR_DIMM0	DR_DIMM2	ANY_DIMM0
SR_DIMM1	DR_DIMM3	ANY_DIMM1
SR_DIMM2	QR_DIMM0	ANY_DIMM2
SR_DIMM3	QR_DIMM1	ANY_DIMM3
DR_DIMM0	QR_DIMM2	

These values can be combined as follows: QR_DIMM0 + ANY_DIMM1

AddrTmg

32 bit address timing value to apply.

ACTION_ODCCTRL

Prototype

```
ACTION_ ODCCTRL (Frequency, DimmConfig, OdcCtrl)
```

Parameters

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

DimmConfig

Mask indicating DIMM configuration. Possible values are:

ANY_	DR_DIMM1	QR_DIMM3
SR_DIMM0	DR_DIMM2	ANY_DIMM0
SR_DIMM1	DR_DIMM3	ANY_DIMM1
SR_DIMM2	QR_DIMM0	ANY_DIMM2
SR_DIMM3	QR_DIMM1	ANY_DIMM3
DR_DIMM0	QR_DIMM2	

These values can be combined as follows: QR_DIMM0 + ANY_DIMM1

OdcCtrl

32 bit output driver compensation control value to apply.

ACTION_SLEWRATE

Prototype

```
ACTION_ SLEWRATE (Frequency, DimmConfig, SlewRate)
```

Parameters

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

DimmConfig

Mask indicating DIMM configuration. Possible values are:

ANY_	DR_DIMM1	QR_DIMM3
SR_DIMM0	DR_DIMM2	ANY_DIMM0
SR_DIMM1	DR_DIMM3	ANY_DIMM1
SR_DIMM2	QR_DIMM0	ANY_DIMM2
SR_DIMM3	QR_DIMM1	ANY_DIMM3
DR_DIMM0	QR_DIMM2	

These values can be combined as follows: QR_DIMM0 + ANY_DIMM1

SlewRate

32 bit slew rate value to apply.

ACTION_SPEEDLIMIT

Prototype

```
  ACTION_ SPEEDLIMIT (DimmConfig, Dimms, Speedlimit15, Speedlimit135,
SpeedLimit125)
```

Parameters

DimmConfig

Mask indicating DIMM configuration. Possible values are:

ANY_	DR_DIMM1	QR_DIMM3
SR_DIMM0	DR_DIMM2	ANY_DIMM0
SR_DIMM1	DR_DIMM3	ANY_DIMM1
SR_DIMM2	QR_DIMM0	ANY_DIMM2
SR_DIMM3	QR_DIMM1	ANY_DIMM3
DR_DIMM0	QR_DIMM2	

These values can be combined as follows: QR_DIMM0 + ANY_DIMM1

Dimms

Mask indicating number of DIMMs in the channel. Possible values are:

NO_DIMM	THREE_DIMM
ONE_DIMM	FOUR_DIMM
TWO_DIMM	ANY_NUM

These values can be combined as follows: ONE_DIMM + TWO_DIMM

Speedlimit15/Speedlimit135/Speedlimit125

Frequencies to apply change. Possible values are:

DDR667_FREQUENCY	DDR1866_FREQUENCY
DDR800_FREQUENCY	DDR2100_FREQUENCY
DDR1066_FREQUENCY	DDR2133_FREQUENCY
DDR1333_FREQUENCY	DDR2400_FREQUENCY
DDR1600_FREQUENCY	

C.5.3 Table Overrides

The following macros provide table driven expert platform specific customizations. Table driven overrides are the preferred mechanism over conditional macros for most processor families.

Table override macros are divided into configuration macros, which specify the platform criteria, and setting override macros, which provide specific setting overrides.

C.5.3.1 Configuration Macros

TBLDRV_CONFIG_TO_OVERRIDE

Prototype

```
TBLDRV_CONFIG_TO_OVERRIDE (DimmPerCH, Frequency, DimmVolt, DimmConfig)
```

Parameters

DimmPerCH

Number of DIMM slots in a channel.

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

DimmVolt

Mask of voltage to apply change. Possible values are:

VOLT1_5_	VOLT1_35_
VOLT1_25_	VOLT_ANY_

These values can be combined as follows: VOLT1_5_+ VOLT1_35_

DimmConfig

Mask indicating DIMM configuration. Possible values are:

ANY_	DR_DIMM1	QR_DIMM3
SR_DIMM0	DR_DIMM2	ANY_DIMM0
SR_DIMM1	DR_DIMM3	ANY_DIMM1
SR_DIMM2	QR_DIMM0	ANY_DIMM2
SR_DIMM3	QR_DIMM1	ANY_DIMM3
DR_DIMM0	QR_DIMM2	

These values can be combined as follows: QR_DIMM0 + ANY_DIMM1

Description

This macro specifies what kind of configuration that applies the override settings. Users are able to state multiple override macros with this configuration macro.

Example:

```
TBLDRV_CONFIG_TO_OVERRIDE (2, DDR1600, VOLT1_5_ + VOLT1_35_,
SR_DIMM0 + DR_DIMM1),
```

```
   TBLDRV_CONFIG_ENTRY_RTTNOM (CS2_ + CS3_, 2),
   TBLDRV_CONFIG_ENTRY_RTTWR (CS2_, 2),
TBLDRV_CONFIG_ENTRY_RTTWR (CS3_, 1),
TBLDRV_CONFIG_ENTRY_ADDRTMG (0x003C3C3C),
TBLDRV_CONFIG_ENTRY_ODCCTRL (0x20112222)
```

Dependencies

The override macros which can be associated with this configuration macro are:

TBLDRV_CONFIG_ENTRY_ODT_RTTNOM,
TBLDRV_CONFIG_ENTRY_ODT_RTTWR,
TBLDRV_CONFIG_ENTRY_ODTPATTERN,
TBLDRV_CONFIG_ENTRY_ADDRTMG,
TBLDRV_CONFIG_ENTRY_ODCCTRL,
TBLDRV_CONFIG_ENTRY_SLOWACCMODE,
TBLDRV_CONFIG_ENTRY_LRDMM_IBT,
TBLDRV_CONFIG_ENTRY_2D_TRAINING

TBLDRV_SPEEDLIMIT_CONFIG_TO_OVERRIDE

Prototype

```
 TBLDRV_SPEEDLIMIT_CONFIG_TO_OVERRIDE (DimmPerCH, Dimms, NumOfSR, NumOfDR,
NumOfQR, NumOfLRDIMM)
```

Parameters

DimmPerCH

Number of DIMM slots in a channel.

Dimms

Number of DIMMs in a channel.

NumOfSR

Number of Single rank DIMMs in a channel

NumOfDR

Number of dual rank DIMMs in a channel

NumOfQR

Number of Quad rank DIMMs in a channel

NumOfLRDIMM

Number of LR-DIMMs (Load reduce DIMMs) in a channel

Description

This configuration macro is only used with memory speed limit override macro, it specifies what kind of configuration that applies the override.

NumOfLRDIMM should be stated as "NA_" in a normal DIMM type platform. NumOfSR, NumOfDR and NumOfQR should be stated as NA_ in a LR-DIMM type platform, vice versa.

Example:

```
TBLDRV_SPEEDLIMIT_CONFIG_TO_OVERRIDE (2, 2, 0, 0, NA_),
TBLDRV_CONFIG_ENTRY_SPEEDLIMIT (DDR1600_FREQUENCY,
DDR1333_FREQUENCY,
DDR1066_FREQUENCY),
```

Dependencies

Only TBLDRV_CONFIG_ENTRY_SPEEDLIMIT override macro can be associated with this configuration macro.

TBLDRV_RC2IBT_CONFIG_TO_OVERRIDE

Prototype

```
TBLDRV_RC2IBT_CONFIG_TO_OVERRIDE (DimmPerCH, Frequency, DimmVolt, DimmConfig,
NumOfReg)
```

Parameters

DimmPerCH

Number of DIMM slots in a channel.

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

DimmVolt

Mask of voltage to apply change. Possible values are:

VOLT1_5_	VOLT1_35_
VOLT1_25_	VOLT_ANY_

These values can be combined as follows: VOLT1_5_+ VOLT1_35_

DimmConfig

Mask indicating DIMM configuration. Possible values are:

ANY_	DR_DIMM1	QR_DIMM3
SR_DIMM0	DR_DIMM2	ANY_DIMM0
SR_DIMM1	DR_DIMM3	ANY_DIMM1
SR_DIMM2	QR_DIMM0	ANY_DIMM2
SR_DIMM3	QR_DIMM1	ANY_DIMM3
DR_DIMM0	QR_DIMM2	

These values can be combined as follows: QR_DIMM0 + ANY_DIMM1

Description

This configuration macro is only used with RC2IBT override macro, it specifies what kind of configuration that applies the override.

Example:

```
TBLDRV_RC2IBT_CONFIG_TO_OVERRIDE  (2, DDR1600, VOLT1_5_ +
VOLT1_35_, SR_DIMM0 + DR_DIMM1, 2),
    TBLDRV_CONFIG_ENTRY_RC2_IBT (DIMM0, 4),
```

Dependencies

Only TBLDRV_CONFIG_ENTRY_RC2_IBT override macro can be associated with this configuration macro.

C.5.3.2 Override Macros

TBLDRV_CONFIG_ENTRY_SPEEDLIMIT

Prototype

```
TBLDRV_CONFIG_ENTRY_SPEEDLIMIT  (SpeedLimit1_5, SpeedLimit1_35, SpeedLimit1_25)
```

Parameters

SpeedLimit1_5/SpeedLimit1_35/SpeedLimit1_25

Frequencies to apply change. Possible values are:

DDR667_FREQUENCY	DDR1866_FREQUENCY
DDR800_FREQUENCY	DDR2100_FREQUENCY
DDR1066_FREQUENCY	DDR2133_FREQUENCY
DDR1333_FREQUENCY	DDR2400_FREQUENCY
DDR1600_FREQUENCY	

Description

This macro specifies memory speed limit value for VDDIO 1.5V, 1.35V and 1.25 V.

Dependencies

This macro must be presented along with TBLDRV_SPEEDLIMIT_CONFIG_TO_OVERRIDE configuration macro.

TBLDRV_CONFIG_ENTRY_ODT_RTTNOM

Prototype

```
TBLDRV_CONFIG_ENTRY_ODT_RTTNOM (TgtCS, RttNom)
```

Parameters

TgtCS

Mask indicating which chip selects are applied to RttNom value override. Possible values are:

CS0_	CS3_	CS6_
CS1_	CS4_	CS7_
CS2_	CS5_	CS_ANY

These values can be combined as follows: CS0_ + CS1_

RttNom

A 8 bits value indicating the nominal termination value. Possible values are:

0 - Disabled	3 - 40 ohms
1 - 60 ohms	4 - 20 ohms
2 - 120 ohms	5 - 30 ohms

Description

This macro specifies the RttNom value override for specific chip selects.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_ODT_RTTWR

Prototype

```
TBLDRV_CONFIG_ENTRY_ODT_RTTWR (TgtCS, RttWr)
```

Parameters

TgtCS

Mask indicating which chip selects are applied to RttNom value override.Possible values are:

CS0_	CS3_	CS6_
CS1_	CS4_	CS7_
CS2_	CS5_	CS_ANY

These values can be combined as follows: CS0_ + CS1_

RttWr

A 8 bits value indicating the write termination value. Possible values are:

0 - Disabled
1 - 60 ohms
2 - 120 ohms

Description

This macro specifies the RttWr value override for specific chip selects.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_ODTPATTERN

Prototype

```
  TBLDRV_CONFIG_ENTRY_ODTPATTERN (RdODTCSHigh, RdODTCSLow, WrODTCSHigh,
WrODTCSLow)
```

Parameters

RdODTCSHigh

A 32 bits value indicating read ODT pattern value for CS[7:4].

RdODTCSLow

A 32 bits value indicating read ODT pattern value for CS[3:0].

WrODTCSHigh

A 32 bits value indicating write ODT pattern value for CS[7:4].

WrODTCSLow

A 32 bits value indicating write ODT pattern value for CS[3:0].

Description

This macro specifies the read/write ODT pattern values to override.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_ADDRTMG

Prototype

```
TBLDRV_CONFIG_ENTRY_ADDRTMG (AddrTmg)
```

Parameters

AddrTmg

A 32 bits value indicating Address/Command Timing value.

Description

This macro specifies the Address/Command Timing value to override.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_ODCCTRL

Prototype

```
TBLDRV_CONFIG_ENTRY_ODCCTRL (OdcCtrl)
```

Parameters

OdcCtrl

A 32 bits value indicating Output Driver Compensation Control (driving strength) value.

Description

This macro specifies the Output Driver Compensation Control value to override.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_SLOWACCMODE

Prototype

TBLDRV_CONFIG_ENTRY_SLOWACCMODE (SlowAccMode)

Parameters

SlowAccMode

A 8 bits value indicating Slow Access Mode value.

Description

This macro specifies the Slow Access Mode value to override.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_RC2_IBT

Prototype

```
TBLDRV_CONFIG_ENTRY_RC2_IBT (TgtDimm, IBT)
```

Parameters

TgtDimm

Mask indicating which DIMMs are applied to RC2 IBT value override. Possible values are:

DIMM0	DIMM3
DIMM1	ALL_DIMMS
DIMM2	

These values can be combined as follows: DIMM0 + DIMM1

IBT

A 8 bits value indicating IBT value to override

Description

This macro specifies the RDIMM IBT value to override.

Dependencies

This macro must be presented along with TBLDRV_RC2IBT_CONFIG_TO_OVERRIDE configuration macro.

TBLDRV_OVERRIDE_MR0_CL

Prototype

```
TBLDRV_OVERRIDE_MR0_CL (RegValOfTcl, MR0CL13, MR0CL0)
```

Parameters

RegValOfTcl

A 8 bits value indicating the value of Tcl bit field value in memory controller register.

MR0CL13

The encoding value of MR0 command CL[3:1] bit field.

MR0CL0

The encoding value of MR0 command CL[0] bit field.

Description

This macro does NOT need to combine with any configuration macro, it specifies the encoding value of MR0 command CL[3:0] bit field to override corresponding to the value of Tcl bit field in memory controller register. Multiple TBLDRV_OVERRIDE_MR0_CL macros are allowed to put together, they override the encoding value of MR0 command CL[3:0] bit field corresponding to various Tcl value in the register.

TBLDRV_OVERRIDE_MR0_WR

Prototype

```
TBLDRV_OVERRIDE_MR0_WR (RegValOfTwr, MR0WR)
```

Parameters

RegValOfTwr

A 8 bits value indicating the value of Twr bit field value in memory controller register.

MR0WR

The encoding value of MR0 command WR[2:0] bit field.

Description

This macro does NOT need to combine with any configuration macro, it specifies the encoding value of MR0 command WR[2:0] bit field to override corresponding to the value of Twr bit field in memory controller register. Multiple TBLDRV_OVERRIDE_MR0_WR macros are allowed to put together, they override the encoding value of MR0 command WR[2:0] bit field corresponding to various Twr value in the register.

TBLDRV_OVERRIDE_RC10_OPSPEED

Prototype

```
TBLDRV_OVERRIDE_RC10_OPSPEED (Frequency, RC10OPSPEED)
```

Parameters

Frequency

Mask of frequencies to apply change. Possible values are:

DDR400	DDR1333
DDR533	DDR1600
DDR667	DDR1866
DDR800	ANY_SPEED
DDR1066	

These values can be combined as follows: DDR400 + DDR533 + DDR667

RC10OPSPEED

The encoding value of RDIMM control word 10 [Operating Speed] bit field.

Description

This macro does NOT need to combine with any configuration macro and is for RDIMM memory type only, it specifies the encoding value of RDIMM RCW10 command [Operating Speed] bit field to override corresponding to memory frequency. Multiple TBLDRV_OVERRIDE_RC10_OPSPEED macros are allowed to put together, they override the encoding value of RDIMM RCW10 command [Operating Speed] bit field corresponding to various memory frequencies.

TBLDRV_CONFIG_ENTRY_LRDMM_IBT

Prototype

```
TBLDRV_CONFIG_ENTRY_LRDMM_IBT (F0RC8, F1RC0, F1RC1, F1RC2)
```

Parameters

F0RC8

A 8 bits value indicating LRDIMM function 0 control word 8 value.

F1RC0

A 8 bits value indicating LRDIMM function 1 control word 0 value.

F1RC1

A 8 bits value indicating LRDIMM function 1 control word 1 value.

F1RC2

A 8 bits value indicating LRDIMM function 0 control word 2 value.

Description

This macro specifies LRDIMM IBT values to override.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_CONFIG_ENTRY_2D_TRAINING

Prototype

```
TBLDRV_CONFIG_ENTRY_2D_TRAINING (Training2dMode)
```

Parameters

Training2dMode

Indicating the enablement of memory 2D training function.

1 - Enable
2 - Disable

Description

This macro is used to force either enable or disable memory 2D training function.

Dependencies

This macro must be presented along with TBLDRV_CONFIG_TO_OVERRIDE configuration macro. And it is able to grouped together with other general override macros using same configuration macro.

TBLDRV_INVALID_CONFIG

Prototype

```
TBLDRV_INVALID_CONFIG
```

Parameters

<none>

Description

This macro specifies an invalid configuration. And, the configuration is stated in one of existing configuration macros.

Dependencies

This macro must be associated with TBLDRV_CONFIG_TO_OVERRIDE, TBLDRV_SPEEDLIMIT_CONFIG_TO_OVERRIDE or TBLDRV_RC2IBT_CONFIG_TO_OVERRIDE configuration macro.

Appendix D Graphics Northbridge Details

Refer to the internal documentation for the latest and most detailed information ("Internal Documentation" on page 25).

D.1 PCIe® Port Descriptor List

These structures provide customization of the processor's PCIe® ports. See processor specific documentation for a description of what PCIe® support, if any, exists on that processor.

Customization data is provided by lists of descriptors. The list is terminated by an item which has DESCRIPTOR_TERMINATE_LIST as its Flags value.

```
typedef struct {
  IN       UINT32               Flags;
  IN       PCIe_ENGINE_DATA     EngineData;
  IN       PCIe_PORT_DATA       Port;
} PCIe_PORT_DESCRIPTOR;
```

Flags

DESCRIPTOR_TERMINATE_LIST - This is the last descriptor in the list.

Engine Data

See "PCIe® Engine Data" on page 331.

Port

Provide customization settings for this PCIe® port.

```
typedef struct  {
  IN       UINT8                  PortPresent;
  IN       UINT8                  ChannelType;
  IN       UINT8                  DeviceNumber;
  IN       UINT8                  FunctionNumber;
  IN       UINT8                  LinkSpeedCapability;
  IN       UINT8                  LinkAspm;
  IN       UINT8                  LinkHotplug;
  IN       UINT8                  ResetId;
  IN       PCIe_PORT_MISC_CONTROL MiscControls;
  IN       APIC_DEVICE_INFO       ApicDeviceInfo;
  IN       PCIE_ENDPOINT_STATUS   EndPointStatus;
  IN       RX_ADAPT_MODE          RxAdaptMode;
} PCIe_PORT_DATA;
```

PortPresent

Specifies the port is enabled. This is specified using the PCIE_PORT_ENABLE enum.

PortDisabled - Disabled

PortEnabled - Enabled

ChannelType

Specifies the type of channel. This is specified using the PCIE_CHANNEL_TYPE enum.

ChannelTypeLowLoss - Low Loss
ChannelTypeHighLoss - High Loss
ChannelTypeMob0db - Mobile 0dB
ChannelTypeMob3db - Mobile 3dB
ChannelTypeExt6db - Extended 6dB
ChannelTypeExt8db - Extended 8dB

DeviceNumber

Specifies the PCI device number for this port.

0 - Use native port device number
N - Use this port device number. See processor specific documentation for details.

FunctionNumber

Reserved.

LinkSpeedCapability

Specifies the link's speed and feature capabilities. This is specified using the PCIE_LINK_SPEED_CAP enum.

PcieGenMaxSupported - The maximum speed and feature set is supported.
PcieGen1 - Gen 1 speeds and features are supported.
PcieGen2 - Gen 2 speeds and features are supported.
PcieGen3 - Gen 3 speeds and features are supported.

LinkAspm

Specifies the link's ASPM control. This is specifed using the PCIE_ASPM_TYPE enum.

AspmDisabled - ASPM is disabled.
AspmL0s - support for the L0s PCIe link state.
AspmL1 - support for the L1 PCIe link state.
AspmL0sL1 - support for both the L0s and L1 PCIe link state.

LinkHotplug

Specifies the link's hot plug support. This is specified using the PCIE_HOTPLUG_TYPE enum.

HotplugDisabled - Hot plug is disabled.
HotplugBasic - Basic hot plug is supported.
HotplugServer - Server hot plug is supported.

HotplugEnhanced - Enhanced hot plug is supported.
HotplugInboard - In board hot plug is supported.

ResetId

Specify the ID which will be passed to the AgesaPcieSlotResetContol callout to reset this port. This generally corresponds to the GPIO which will be used by the host BIOS to control this port's reset. Ports may share a reset ID, if they are controlled together by the platform.

MiscControls

Miscellaneous other controls for port test and debug.

ApicDeviceInfo

IOAPIC device and interrupt programming information.

EndPointStatus

Provides additional status information about the end point device.

EndpointDetect - Indicates that AGESA should attempt to detect endpoint presence

EndpointNoPresent - Indicates that there is an alternative way to determine if the endpoint is present or not (such as a GPIO) and that method has indicated that there is no device present. AGESA will not attempt to detect a device in this port

RxAdaptMode

Defines the default configuration for RxAdaptMode used for Gen3 PCIe configuration. This parameter structure generally applies only to Gen3 capable slots and devices.

```
typedef struct {
      IN    UINT8       GroupMap;
      IN    UINT8       Swizzle;
      IN    UINT8       BridgeInit;
} APIC_DEVICE_INFO;
```

GroupMap

The four virtual PCIe® interrupts, INTx, may be mapped to a group of four entries in the redirection table.

0 - Use the recommended settings (default). The other members of this structure will be ignored.
1 - Interrupts are mapped to Group 0 which is interrupts 0 through 3 of the redirection table.
2 - Interrupts are mapped to Group 1 which is interrupts 4 through 7 of the redirection table.

3 - Interrupts are mapped to Group 2 which is interrupts 8 through 11 of the redirection table.
4 - Interrupts are mapped to Group 3 which is interrupts 12 through 15 of the redirection table.
5- Interrupts are mapped to Group 4 which is interrupts 16 through 19 of the redirection table.
6 - Interrupts are mapped to Group 5 which is interrupts 20 through 23 of the redirection table.
7 - Interrupts are mapped to Group 6 which is interrupts 24 through 27 of the redirection table.
8 - Interrupts are mapped to Group 7 which is interrupts 28 through 31 of the redirection table.

Swizzle

Within the interrupt group, the virtual PCIe® interrupts, INTx, may be swizzled.

0 - Use swizzle ABCD (for example, if the group is 1 then INTA is level 0, INTB is level 1, INTC is level 2, and INTD is level 3).
1 - Use swizzle BCDA.
2 - Use swizzle CDAB.
3 - Use swizzle DABC.

BridgeInt

The PCIe® bridge will use this interrupt level for the bridge Message Signalled Interrupt (MSI). Specify an available interrupt level in the redirection table, from 0 to 31.

```
typedef struct {
  IN  BOOLEAN           InitOffsetCancellation;
  IN  UINT8             DFEControl;
  IN  UINT8             LEQControl;
  IN  BOOLEAN           DynamicOffsetCalibration;
  IN  BOOLEAN           FOMCalculation;
  IN  BOOLEAN           PIOffsetCalibration;
} RX_ADAPT_MODE;
```

See the enum definition in the AGESA.H file for permitted values.

InitOffsetCancellation

Defines whether Initial Offset calibration will be enabled or not.

DFEControl

Defines the configuration of the LEQ Control.

DynamicOffsetCalibration

Defines whether Dynamic Offset Calibration is enabled or not.

FOMCalculation

Defines whether FOM Calculation will be enabled or not.

PIOffsetCalibration

Defines whether PI Offset Calibration will be enabled or not

D.2 DDI Link Descriptor List

These structures provide customization of the processor's DDI ports. See processor specific documentation for a description of what DDI support, if any, exists on that processor.

Customization data is provided by lists of descriptors. The list is terminated by an item which has DESCRIPTOR_TERMINATE_LIST as its Flags value.

```
typedef struct {
  IN       UINT32               Flags;
  IN       PCIe_ENGINE_DATA     EngineData;
  IN       PCIe_DDI_DATA        Ddi;
} PCIe_DDI_DESCRIPTOR;
```

Flags

`DESCRIPTOR_TERMINATE_LIST` - This is the last descriptor in the list.

Engine Data

See "PCIe® Engine Data" on page 331.

Ddi

Provide customization settings for this Ddi port.

```
typedef struct  {
  IN       UINT8                  ConnectorType;
  IN       UINT8                  AuxIndex;
  IN       UINT8                  HdpIndex;
  IN       CONN_CHANNEL_MAPPING   Mapping[2];
  IN       UINT8                  LanePnInvert;
  IN       UINT8                  Flags;
} PCIe_DDI_DATA;
```

ConnectorType

Specifies the display connector type. Connector types are provided by the PCIE_CONNECTOR_TYPE enum.

ConnectorTypeDP - DP
ConnectorTypeEDP - eDP
ConnectorTypeSingleLinkDVI - Single Link DVI
ConnectorTypeDualLinkDVI - Dual Link DVI
ConnectorTypeHDMI - HDMI™
ConnectorTypeDpToVga - Travis DP to VGA
ConnectorTypeDpToLvds - Travis DP to LVDS
ConnectorTypeNutmegDpToVga - Nutmeg DP to VGA
ConnectorTypeSingleLinkDviI - Single Link DVI-I
ConnectorTypeCrt - Native CRT
ConnectorTypeLvds - Native LVDS
ConnectorTypeEDPToLvds - common eDP to LVDS
ConnectorTypeEDPToLvdsSwInit - eDP to LVDS, with specifc software initialization steps.
ConnectorTypeAutoDetect - Provides automatic detection for EDID display panels when connected via Native LVDS, eDP, or Travis DP to LVDS.

AuxIndex

Specifies the AUX line used. AUX lines are specified using the PCIE_AUX_TYPE enum: AUX1, AUX2, AUX 3, AUX, AUX5, and AUX6.

HdpIndex

Specifies the HDP pin for this port. HDP pins are specified using the PCIE_HDP_TYPE enum: HDP1, HDP2, HDP3, HDP4, HDP5, and HDP6.

Mapping

Specifies the mapping of lanes to connector pins. Mapping [0] defines the mapping for the four lanes beginning at PCIe_ENGINE_DATA.StartLane. Mapping [1] defines the mapping for the four lanes ending at PCIe_ENGINE_DATA.EndLane. Lane mapping values are:

0 - Map to lane 0.
1 - Map to lane 1.
2 - Map to lane 2.
3 - Map to lane 3.

LanePnInvert

Specifies whether to invert the state of P and N for each lane.

0 - Do not invert (default).
1 - Invert P and N on this lane.

Flags

Specifies capability flags for the DDI link. By default, no special capability limits are assumed. Each flag can be used to limit that capability.

DDI_DATA_FLAGS_DP1_1_ONLY - Limit the link to DP 1.1 capable.
EXT_DISPLAY_PATH_CAPS_DP_REPEATER - Apply special settings for certain hardware implementations.

D.3 PCIe® Engine Data

This structure is used for configuring both PCIe® and DDI ports.

```
typedef struct {
  IN       UINT8               EngineType;
  IN       UINT16              StartLane;
  IN       UINT16              EndLane;
} PCIe_ENGINE_DATA;
```

EngineType

Indicates the port type which this customizes.

0 - Ignore this item
1 - PCIe® Port
2 - DDI Port

StartLane

The starting Lane ID for this port. Normally, that is the lowest Lane ID, but it can be the highest Lane ID when the lanes are reversed. See processor specific documentation for details.

EndLane

The ending Lane ID for this port. Normally that is the highest Lane ID, but it can be the lowest Lane ID when the lanes are reversed. See processor specific documentation for details.

D.4 IOMMU Exclusion Range Descriptor

Platform BIOS may provide IOMMU exclusion ranges for I/O virtualization as follows.

```
typedef struct {
  IN       UINT16     Bus       :8;
  IN       UINT16     Device    :5;
  IN       UINT16     Function  :3;
} IOMMU_REQUESTOR_ID;

typedef struct {
  IN       UINT32               Flags;
  IN       IOMMU_REQUESTOR_ID   RequestorIdStart;
```

```
  IN       IOMMU_REQUESTOR_ID   RequestorIdEnd;
  IN       UINT64               RangeBaseAddress;
  IN       UINT64               RangeLength;
} IOMMU_EXCLUSION_RANGE_DESCRIPTOR;
```

Flags

This descriptor may be ignored or may terminate the list.

RequestorIdStart

The exclusion range is requested for the inclusive device set beginning with the requestor start ID.

RequestorIdEnd

The end of the requestor ID range.

RangeBaseAddress

The allocate the exclusion range beginning at this physical address.

RangeLength

The length of the exclusion region in bytes.

D.5 ACPI ASL Library

Platform BIOS may include ACPI ASL methods for providing run-time features, for example, PCIe® Speed Power Policy (PSPP) or PCIe® hot plug. The ACPI ASL library provides support for these platform BIOS ACPI ASL methods by providing a library of supporting ASL methods which can be invoked by the platform BIOS ASL code.

Use of the ACPI ASL Library requires that PCIe® MMIO base address and size are configured. See “BLDCFG_PCI_MMIO_BASE” on page 182 and “BLDCFG_PCI_MMIO_SIZE” on page 182.

When using the ACPI ASL Library, the following names should be considered reserved in _SB scope: ALIB; ADxx, where xx is 00 - 99; Ayyy, where yyy is 000 - 999.

ACPI ASL Library support is invoked using the ALIB method. This method is in _SB scope and provides multiple services as indicated by a function code.

```
ALIB (Arg0, Arg1)
```

Arg0

Provide the desired function code. Valid function codes are specified below.

Arg1

A parameter buffer which provides inputs to the method and contains outputs from the method. The inputs and outputs are specific to each function. The buffer is 256 bytes.

These are the supported functions. For each function, the Arg1 buffer parameters are described for input and output.

Report AC/DC State - Function 1.

The Platform BIOS should Report AC/DC State to report boot up power source as AC or DC. This function could be performed from _SB.PCI0._INI or from _SB._INI, for example.

Input:

```
WORD Size
BYTE AC/DC State
```

Size

The size in bytes, 3.

AC/DC State

Indicate the current power source type.

0 - Current state is AC power.
1 - Current state is DC power.

Output: None.

PCIe®Performance Request - Function 2.

Register a specific performance request for a specific client PCIe® device, or remove the previous request.

Input:

```
WORD Size
WORD Client ID
WORD Valid Flags
WORD Flags
BYTE Request Type
BYTE Performance Request
```

Size

The size in bytes, 10.

Client ID

Provides client device PCI config address, Bus in bits [15:8], Device in bits [7:3], and Function in bits [2:0].

Valid Flags

Indicates which bits in Flags are valid. For each bit:

0 - Flag bit is not valid.
1 - Flag bit is valid.

Flags

bit 0 - Advertise capabilities.

bit 1 - Wait for completion.

bits [31:2] - Reserved, must be zero.

Request Type

Specifies how to interpret the performance level in Performance Request.

1 - PCIe® Link Speed

Performance Request

0 - Remove request.
1 - Force low power mode.
2 - Performance level 1 (PCIe® Gen1 speed).
3 - Performance level 2 (PCIe® Gen2 speed).
4 - Performance level 3 (PCIe® Gen3 speed).

Output:

```
WORD Size
Byte Return Value
```

Size

The size in bytes, 3.

Return Value

1 - Request refused.
2 - Request complete.
3 - Request in progress.

PSPP Start/Stop Management Request - Function 3.

Start, or stop, PSPP policy management.

Input:

```
WORD Size
BYTE Policy
```

Size

The size in bytes, 3.

Policy

0 - Stop PSPP policy management.
1 - Start PSPP policy management.

Output:

```
WORD Size
BYTE Status
```

Size

The size in bytes, 3.

Status

0 - Request completed successfully.
1 - Request unsupported.
5 - Request error, AGESA_ERROR severity.

Set PCIe® Bus Width - Function 4.

Set or reset power gating based on the number of requested PCIe® lanes.

Input:

```
WORD Size
WORD Client ID
BYTE Lanes
```

Size

The size in bytes, 5.

Client ID

Provides client device PCI config address, Bus in bits [15:8], Device in bits [7:3], and Function in bits [2:0].

Lanes

Specifies the number of lanes to be active.

Output:

```
WORD Size
BYTE Lanes
```

Size

The size in bytes, 3.

Lanes

The number of active lanes. May be more or less than the number requested.

ALIB Init - Function 5.

Initialize the ACPI ASL library. Recommend this is called from _SB.PCI0._INI.

Inputs: None.

Outputs: None.

PCIe® Port Hot Plug Request - Function 6.

Handle device insert and eject for a specific PCIe® port. This should be called from the SCI handler for the “PRSNT#” signal on the PCIe® port.

Input:

```
WORD Size
WORD Port ID
BYTE State
```

Size

The size in bytes, 5.

Port ID

Provides port device PCI config address, Bus in bits [15:8], Device in bits [7:3], and Function in bits [2:0].

State

0 - Handle device eject.
1 - Handle device insert.

Output:

```
WORD Size
BYTE Status
BYTE Device Status
```

Size

The size in bytes, 4.

Status

0 - Request completed successfully.
1 - Request unsupported.
5 - Request error, AGESA_ERROR severity.

Device Status

0 - Device not present.
1 - Device present.

Report Dock/Undock State - Function A

The Platform BIOS should Report Dock/Undock State to report attachment to a dock that provides supplemental cooling or thermal capabilities. This function could be performed from _SB.PCI0._INI or from _SB._INI, for example. This function is independent of the AC/DC function.

Input:

```
WORD  Size
BYTE  Dock/Undock State
```

Size -

The size in bytes.

Dock/Undock State -

Indicate the current power source type.

0 - Current state is docked
1 - Current state is not docked

Output: None.

Report Battery Status - Function B

The Platform BIOS should report battery status information for management of power and performance states while operating in DC mode. This information should be reported periodically as the battery status is measured by the system and should be reported independently for each battery in the system.

Input:

```
WORD  Size
BYTE  Battery ID
BYTE  Power Unit
DWORD Battery Total Capacity
DWORD Battery Remaining Capacity
DWORD Battery Voltage
```

Size -

The size in bytes, 16

Battery ID -

The ID of the battery being reported

Power Unit -

Indicates the current power source type.

0 – Battery capacity is in mAh
1 – Battery capacity is in mWh

Battery Total Capacity -

Indicates the total capacity of the battery when fully charged, in the units specified by Power Unit. This should match the last reported value by the ACPI _BIF object.

Battery Remaining Capacity -

Indicates the remaining capacity of the battery, in the units specified by Power Unit.

Battery Voltage -

Indicates the voltage currently provided by the battery, in mV. For systems reporting the remaining capacity in mAh the voltage is used to calculate the battery capacity in mWh units.

Oubtput: None.

D.5.1 Optional ACPI Callout Method

The ACPI ASL Library provides for calling an optional ACPI ASL method to control PCIe® slot reset during hot plug. The platform BIOS should implement this method in order to support PCIe® hot plug. To implement the method, define:

```
\_SB.ALIC (Arg0, Arg1)
```

Arg0

The Bus, Device, Function of the slot's PCI address.

Function in bits 2:0
Device in bits 7:3
Bus in bits 15:8

Arg1

Indicates whether to assert or deassert reset to the slot.

0 - Assert reset.
1 - Deassert reset.

D.5.2 AWAK and APTS Methods

The ACPI ASL library provides methods to assist the _WAK and _PTS ACPI control methods in waking the system or preparing the system to sleep.

```
AWAK (Arg0)
```

Arg0

An integer indicating the sleep state from which the system is waking. This should be the same value passed to the _WAK method. The S1 sleep state is represented as 1, S2 as 2, etc.

AWAK is a method which provides processor specific configuration after the system wakes from a sleep state. The host environment should invoke AWAK from its _WAK method. AWAK does not return a value.

```
APTS (Arg0)
```

Arg0

An integer indicating the sleep state to which the system is preparing to enter. This should be the same value passed to the _PTS method. The S1 sleep state is represented as 1, S2 as 2, etc.

APTS is a method which provides processor specific configuration as the system prepares to enter a sleep state. The host environment should invoke APTS from its _PTS method. APTS does not return a value.