AMIBIOS®8

American Megatrends

www.ami.com

Introduction to AMIBIOS8™

Overview of key features in the latest AMIBIOS™

Version 1.7 – June 16, 2009

American Megatrends, Inc.

5555 Oakbrook Parkway

Suite 200

Norcross, GA 30093

Revision Information

Date	Rev	Description of Changes
2001-11-05	1.0	First public release
2001-11-06	1.01	Format change on front page
2002-02-08	1.10	Fixed trademark issues. Added overview.
2002-03-25	1.20	Changed to new style template. Added notebook BIOS info to Section 5. Removed references to PARTIES. Added more VeB screenshots (BIOS Parameters & tools).
2005-05-12	1.21	Updated template
2005-04-14	1.30	Updated for 8.00.12
2007-10-29	1.40	Updated to 8.00.15. Updated to new documentation standard. Updated copyright dates. Removing “PC 2001” references. Add RSC information. Removed Section 6.
2008-01-15	1.50	Updated Server platform eModule list. Updated document header & copyright dates for 2008. Updated cover page.
2008-05-05	1.60	Updated corporate headquarters address. Added references to Blu-ray & HD DVD in footnotes. Added EC to Section 5.4.
2009-06-16	1.70	Added AMI Debug Rx information (Section 2.8). Reformatted pictures to remove dead space in document layout.

Table of Contents

1 Introduction

1.1 Purpose of this document

This document is an overview of AMIBIOS8™, specifically improvements to the core architecture and development toolkit. AMIBIOS8™ incorporates the proven support of previous AMIBIOS® releases, along with support for the latest technology. AMIBIOS8™ development focused on these key areas:

- Ease porting to new chipsets & platforms
- Reduce the effort of maintaining BIOS code
- Increase modularity so unwanted components can be removed
- Make AMIBIOS® an appealing solution for non-traditional x86 platforms
- Decrease "total time to market" when developing BIOS solutions

1.2 Features of AMIBIOS8™

AMIBIOS8 features a smaller footprint, fast boot times, efficient power management features and support for a wide set of features, including:

- Power Management: ACPI, Embedded Controller, Intel® SpeedStep® and Enhanced SpeedStep®, AMD PowerNOW! and Cool n'Quiet
- BBS Boot Options: SATA, IDE, USB, LAN, Floppy
- Trust & Security: TCG/TPM, Intel® TXT
- Virtualization: Intel® VT, AMD-V™
- Platform Management: Intel® vPro™, Intel® AMT, IPMI, SMBIOS, ASF, Serial Console Redirection
- Expansion Bus: USB, PCI, PCI-X, PCIe, ExpressCard, HTX

There are several key aspects of AMIBIOS8™ designed to improve BIOS development:

- BIOS architecture enhancements
- Table Driven POST & Single Link Architecture BIOS
- Modular BIOS Components (eModules)
- Separation of silicon support code (Chipset, CPU, Flash, Super I/O)
- Support for Legacy Free platforms
- Visual eBIOS (VeB) development environment for Microsoft Windows
- Project management, including tree organization and source control
- Built in editor with context sensitive help
- Graphical representation of PCI IRQ Routing
- AMI Debug Rx for quick access to POST Checkpoints via USB
- AMI Debug for AMIBIOS8 for BIOS source-level debug capabilities
- Simplified directory structure, organized by component type

2 Enhancements to BIOS Architecture

2.1 Table Driven Post (TDP)

In AMIBIOS8™ the organization of BIOS Power-On Self Test (POST) has been broken down into a series of independent functions. The top level of POST utilizes two components:

1. A table that contains a pointer to each of these functions
2. Code that calls each function in the table

This architectural change has three major advantages:

1. The control flow of POST is very easy to understand and debug
2. Custom routines can easily be added to the table to execute OEM specific code at any point in POST
3. Core updates or bug fixes can easily be distributed and integrated into existing BIOS projects without disturbing the custom routines that have been added at various points in POST

Optional BIOS components such as USB and ACPI may be added to an AMIBIOS8™ project in the form of eModules. Each eModule is described using System Description Language (SDL). The eModule's SDL file describes what component routines must get called during POST, and at what point during POST they should be executed. Calls to the component's various entry points are automatically added to the TDP table.

Table Drive POST is one of the key features that enables AMIBIOS8™ to exceed fast boot benchmarks in Microsoft logo requirements. Fast POST is discussed in Section 3.1 of this document.

2.2 Single Link Architecture BIOS (SLAB)

Predecessors of AMIBIOS8™ utilized a code module system that executed BIOS code from multiple segments, each disconnected from the other. This required extra code to allow data sharing between segments and an intra-module calling mechanism.

AMIBIOS8™ uses the Single Link Architecture BIOS (SLAB) model to overcome these limitations. In AMIBIOS8™ most BIOS modules are linked together at build time. Each binary module is now a code segment in a single large executable image. Any module may call or refer to data in any other module by simply using an **extern far**. The following modules are linked to form the single executable image:

- POST
- Runtime
- Device Initialization Manager (DIM)
- System Management Mode (SMI)
- INT 10 & INT 13
- POST Memory Manager (PMM)
- Uncompression Interface
- BIOS Setup Server

> *Note: The BIOS Boot Block remains a separate module, which is responsible for decompressing the BIOS image from ROM after memory initialization.*

A beneficial side effect of SLAB is the ability to generate a map file at BIOS build time that contains the exact segment and offset of every public (code and data) that is defined in all of the above modules. This map file is very valuable for debugging.

SLAB also allows for a reduction in the number of files needed to build the BIOS. Consider a hypothetical new technology called 'XYZ'. Adding support for 'XYZ' in the previous BIOS model would require the creation of several new files:

- RUNXYZ.ASM (for runtime functions)
- POSTXYZ.ASM (for initialization functions)
- SMIXYZ.ASM (code that resides in SMM)

Under AMIBIOS8™, SLAB allows this new technology to be supported by adding a single file - XYZ.ASM. This single file would contain several different segments:

- RUNTIME segment for runtime code
- POST segment for init code
- SMI segment for SMI related code (APM, ACPI, USB, etc.)

2.3 eModules

In AMIBIOS8™ many optional components are packaged into eModules. An eModule is a package of files that may easily be added to or removed from a BIOS project. Examples of eModules that are available in AMIBIOS8™ are:

- System Management Interrupt (SMI) Interface
- Advanced Configuration & Power Interface (ACPI)
- Advanced Power Management (APM)
- Universal Serial Bus (USB)
- BIOS Debugger
- Console Redirection

All eModules have the following characteristics:

- Consist of a group of files, some of which are core, some chipset, and some board or OEM related
- Include a Component Information Format (CIF) file that lists all of the eModule's source files
- Include a MAK file that completely controls the building of the eModule's source, no other MAK file is modified when an eModule is added to a project
- Include a System Description Language (SDL) file that contains any make flags used to compile the eModule into the BIOS project. The SDL file also indicates what entry points need to get called during POST.

If a particular BIOS project does not need the functionality provided by a given eModule, then the eModule's code (including MAK, SDL, and CIF files) can be completely removed from the BIOS project. This simplifies the source tree for

embedded BIOS projects and other non-traditional PC projects that do not require many BIOS features.

AMIBIOS8™ documentation includes a design guide that describes how to structure BIOS code for a new technology so that it can be packaged and released as an eModule.

2.4 North Bridge / South Bridge Code Separation

Recent trends in chipset design have led to the concept of "mix and match" north and south bridge components. Often a north bridge from a chipset manufacturer can be used with a number of south bridge components from that manufacturer or even from a different supplier. Previous AMIBIOS® releases used a 'monolithic' chipset layer – north bridge and south bridge code is intertwined and interdependent. This made it difficult to support a "mix and match" bridge model.

In AMIBIOS8™, BIOS code to support the north bridge and south bridge is contained in separate files, and released as separate components. This simplifies the maintenance of code; bugs can be fixed once, then propagated to several mixtures of system bridges.

2.5 Modular SMI Support

In previous AMIBIOS® releases, SMI support and Advanced Power Management (APM) support were intertwined and inseparable. While today's desktop systems require support for SMI, APM is becoming obsolete (replaced by APCI).

In AMIBIOS8™, the base SMI code has been separated into an eModule. An AMIBIOS8™ based BIOS can have SMI support without having any APM support code. This results in a ROM image space savings, a large amount of simplification in the source code, and eases debugging considerably. SMI code and data are separated into multiple logical segments. The SMI module with the help of SLAB provides optimum allocation of code and data areas in the available SMRAM at build time. Because of this efficient allocation, more free space in SMRAM is available for OEM code. AMIBIOS8™'s SMI module can support any combination of up to 16 logical processors.

If a platform requires APM support under AMIBIOS8™, it may be added using the AMIBIOS8™ APM eModule.

2.6 Legacy Free

The concept of "legacy free" computing revolves around the removal of older technologies to improve the overall user experience. Legacy free design guidelines are available from Microsoft and Intel.

There are two levels of legacy free support in AMIBIOS8:

1. Legacy Free
   - No ISA devices or ISA expansion slots
   - No floppy disk controller

- No 8042 style keyboard controller (replaced by USB)
- No serial port hardware at "legacy" addresses (COM1/2/3/4)
- No parallel port hardware at "legacy" addresses (LPT1/2/3)
- No game/joystick port (15-pin analog style)

2. Legacy Reduced
   - Same as legacy free, except system has keyboard controller

Although previous AMIBIOS® releases had the ability to support legacy free platforms, it required a significant amount of customization to run on true legacy free systems (no keyboard controller). In AMIBIOS8™, all code related to legacy devices has been isolated and can easily be excluded or included in any BIOS build.

2.7 CPU Module

The CPU support module for AMIBIOS8™ has been completely rewritten to handle new CPU architectures, including products from Intel, AMD and VIA. All CPU related code has been isolated into a single module, easily be upgraded to support new CPU features as well as higher clock speeds. The AMIBIOS8™ CPU module includes advanced cache handling in the initialization code, greatly improving boot times.

2.8 AMI Debug Rx

AMI Debug Rx is the first of its kind: a low-cost debug tool built around the debug port feature common to today's USB 2.0 EHCI controllers. Perfect for today's embedded & netbook platforms, this product is targeted to power users, quality assurance labs & service technicians. Diagnosing small form factor platforms with AMI Debug Rx is non-intrusive, allowing technicians to access checkpoints without opening the case.

- Replaces "POST Checkpoint Card" in development & diagnostics
- No need for proprietary debug headers … no need to open the case
- Records checkpoints & timing data for measuring boot performance
- Session data can be stored to one of four "sessions" for later review
- Translated checkpoint values to descriptive text: based on AMIBIOS8, Aptio 4.x or a user-provided custom checkpoint table
- Quickly ported into existing AMIBIOS8 and Aptio 4.x BIOS projects
- Works with AMI Debug for AMIBIOS8 for source-level BIOS debugging

AMI Debug Rx is scheduled to ship in the second half of 2009. For more information, contact an AMI Sales Associate.

3 Key Features

AMIBIOS8™ builds upon the current level of industry support in the AMIBIOS®. Several new features have been added to AMIBIOS8™ to support the latest platforms.

3.1 Fast POST

Requirements from Microsoft and have pushed the industry to improve boot time. AMIBIOS8™ meets these "Fast POST" requirements, allowing POST execution in a matter of seconds. With a 7200 RPM IDE hard disk drive, an AMIBIOS8™ desktop system can easily boot under 10 seconds[1].

Architectural enhancements such as Table Driven POST and Single Link Architecture have further improved boot time. Excluding eModules that are not required by a particular platform can also optimize boot time.

3.2 USB Mass Storage

USB Mass Storage support was completed for previous AMIBIOS® releases, but was only available as an enhancement kit. AMIBIOS8™ includes integrated support for USB mass storage devices[2], including:

- USB Hard Disk Drive
- USB Flash Media (memory cards & "thumb drives")
- USB DVD±R, DVD±RW, CD-ROM & CD-RW[3]
- USB Floppy, Zip & LS-120/LS-240 "Superdisk"

3.3 Headless Operation

AMIBIOS8™ adds two components to improve manageability of headless systems:

1. Console Redirection – BIOS POST display, including test-based option ROM menus, are accessible using the system serial port. Input/output are accessible via a standard terminal program. AMIBIOS8™ console redirection supports ANSI, VT100 and the Microsoft VT-100+ standard.
2. Remote Flash – Using a standard terminal program with XMODEM support, BIOS flash can be performed via a standard serial port.

Console redirection and remote flash update over RS-232 were implemented in previous AMIBIOS® releases as enhancement kits. AMIBIOS8™ includes these features in the form of an eModule that can be enabled by a simple build switch.

---

[1] System configuration does affect boot time. Memory size, CPU speed, attached USB devices and add-on devices with Option ROMs are all factors in POST execution time.

[2] Not all USB-based storage devices are USB Mass Storage compliant. Please refer to the USB Mass Storage class specifications, which are available at http://www.usb.org/

[3] AMIBIOS8 only provides read-only support from Blu-ray Disc, HD DVD, DVD±RW & CD-RW.

3.4 48-bit LBA Support for IDE Drives

Previous AMIBIOS® releases use the standard 28-bit LBA addressing to access IDE drives. This method cannot handle drives larger than 137GB. New ATA/IDE hard disk drives utilize a new 48-bit addressing scheme. This new standard has the ability to address drives over 144,000 terabytes in size. AMIBIOS8™ offers support for the new LBA addressing scheme and therefore can handle drives larger than 137G.

3.5 ACPI 3.0

The *Advanced Configuration and Power Interface Specification, Revision 3.0* describes the structures and mechanisms necessary to design operating system-directed power management and make advanced configuration architectures possible. ACPI applies to all classes of computers.

Revision 3.0 of the ACPI specification adds numerous multiprocessor workstation and server-related enhancements. The AMIBIOS8 ACPI eModule includes the following:

- General configuration enhancements.
- Inter-processor power, performance, and throttling state dependency support
- Support for >256 processors
- NUMA Distancing support
- SRAT, SLIT & SPMI tables
- PCI Express support
- SATA support
- Ambient Light Sensor and User Presence device support
- Thermal model extended beyond processor-centric support
- Intel High Definition Audio (HAD)
- Multicore processor power management (C-states and P-states)
- Hotplug
- Battery recalibration
- New thermal requirements
- Wake/Sleep/Wake sequencing

4 Enhancements to Development Environment

4.1 Visual eBIOS Development Environment (VeB)

AMIBIOS8™ includes the Visual eBIOS (VeB) integrated development environment (IDE). While VeB simplifies BIOS porting and project management tasks, engineers may still use the familiar DOS environment for BIOS development. VeB is a comprehensive tool that is used during all aspects of BIOS development.

4.1.1 Project Management

VeB includes tools that allow an engineer to quickly define a new project based on a global component library. VeB provides a new project wizard that assists the engineer in pulling together the BIOS components needed to form the project. VeB prompts for the following components:

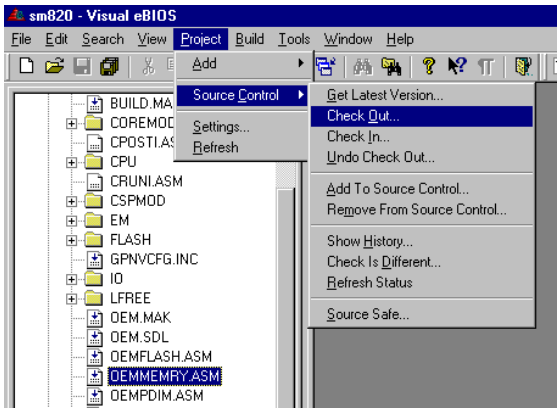
- AMIBIOS8™ Core
- Chipset Support Package (CSP) for Northbridge/Southbridge (or MCH/ICH)
- Board Support Package (BSP)
- Super I/O Support Module
- Flash Support Module
- CPU Family Module

Once a project has been created, additional components may be added or removed at any time.

4.1.2 Source Control

VeB contains integrated source control, with support for multiple formats:

- Visual Source Safe
- ClearCase
- PVCS
- Dimensions
- Subversion
- AMI Remote Source Control (RSC), providing customer access to AMI source databases 24/7/365

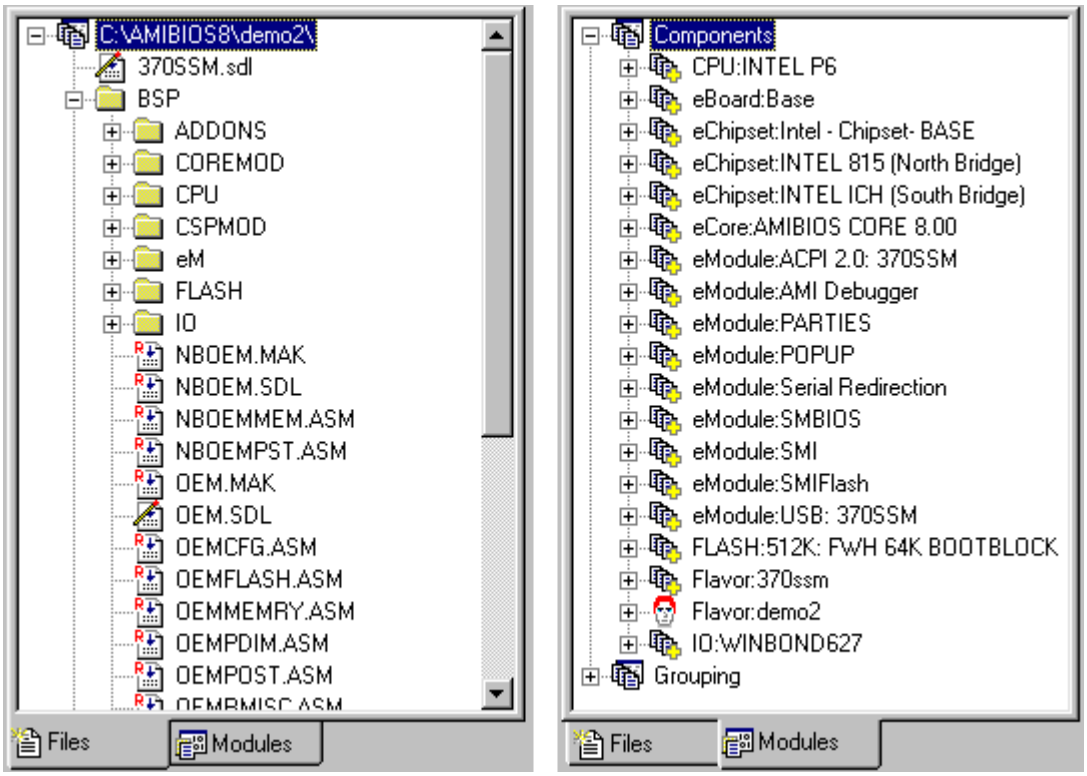


All standard source control operations can be done directly from VeB including check out, check in, get latest, add, remove, history, and difference. VeB provides a more convenient way to manage source.

4.1.3 Source File Views

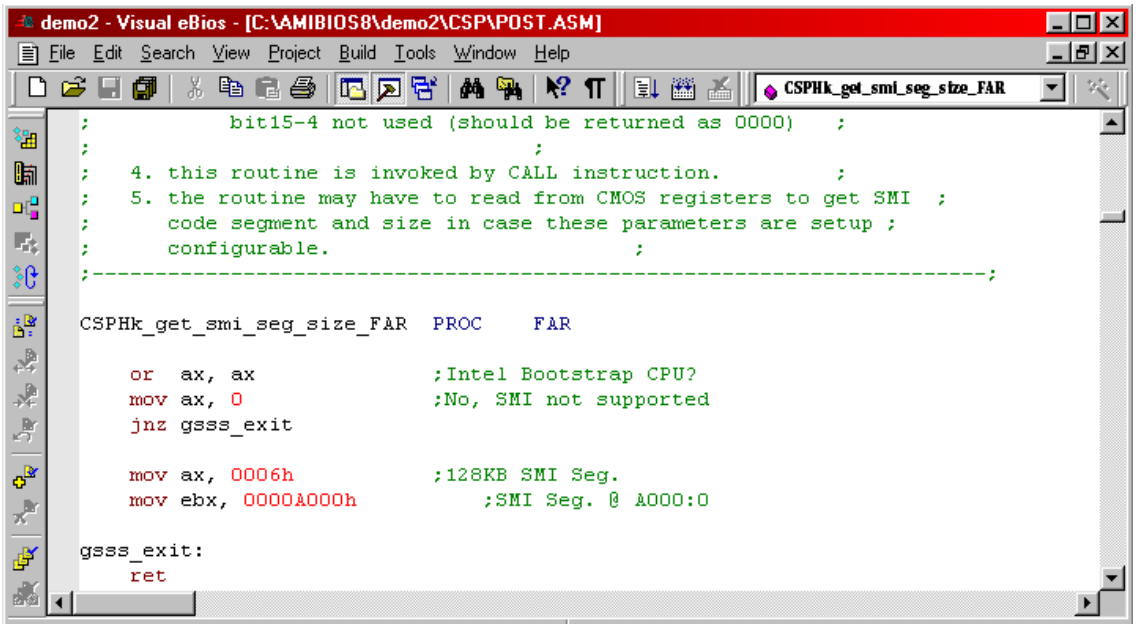
VeB allows the engineer to view a project two different ways:

1. A tree of files organized exactly as they are in the project's directory tree.
2. A tree of files organized by BIOS component.



This very powerful feature allows the engineer to quickly navigate through the source and find the code easily. The component view also gives a better understanding to what is happening when BIOS components are added to or removed from a project.

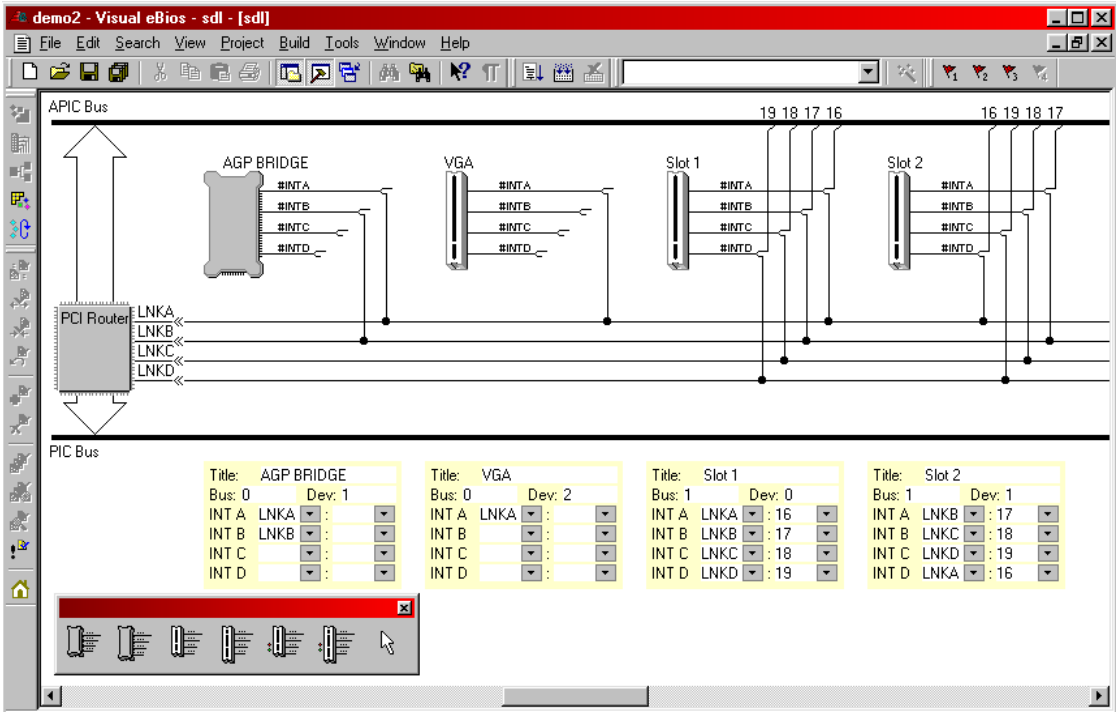
4.1.4 Built in Editor



VeB includes a built in editor. VeB's editor supports syntax highlighting and many other features available with popular source code editors. The user may configure VeB to use its internal editor or to launch an external editor of the user's choice when opening a file. VeB's editor may be customized to use any hot key combination for common editing tasks. This allows the editor to emulate other popular editors.

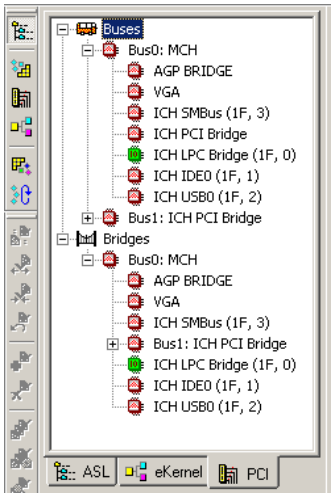
4.1.5 IRQ Routing Wizard

VeB includes the graphical IRQ Routing Wizard. This gives engineers working with schematics a more intuitive way to customize the IRQ routing for a platform. This screen also allows hardware engineers to verify the IRQ routing information without having to understand the syntax of IRQ routing tables.



The information that is gathered by the IRQ Routing Wizard is used to generate several sections of source code:

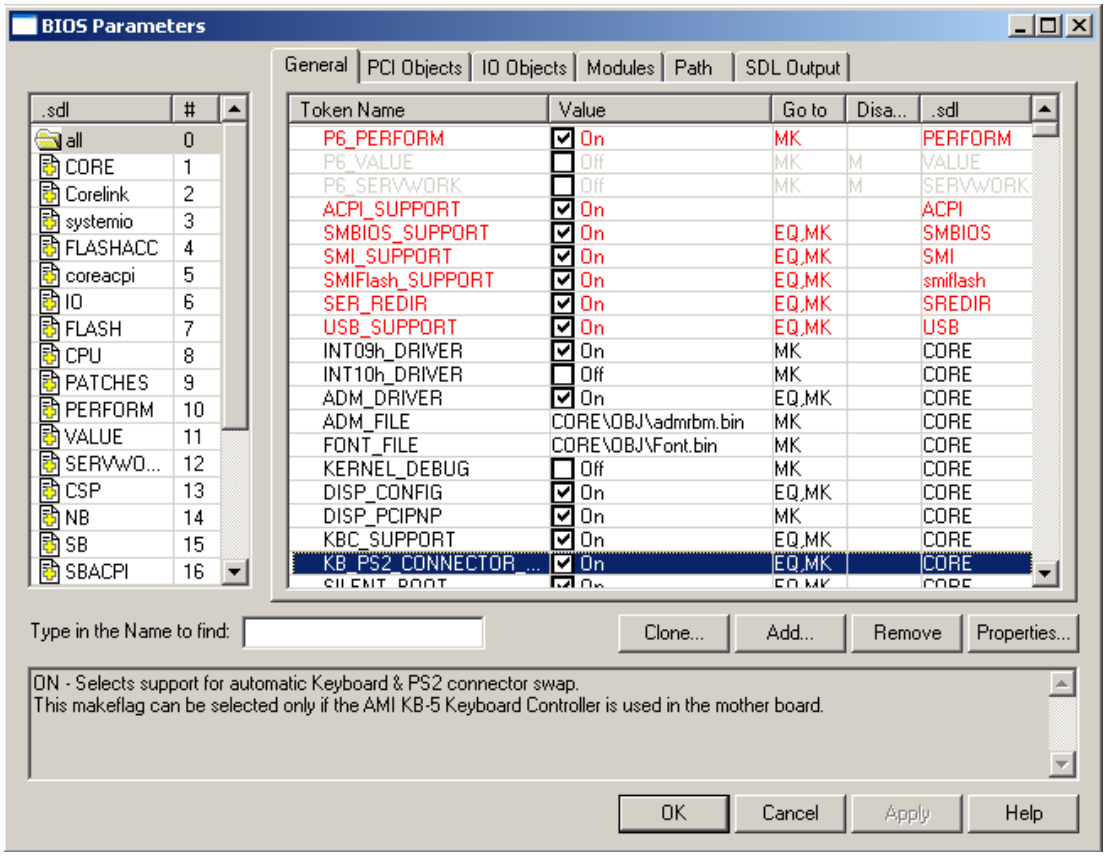
- PCI BIOS IRQ routing table
- Microsoft IRQ routing table
- ACPI IRQ routing information
- MultiProcessor (MP) table IRQ information



VeB also allows IRQ information to be organized by bus hierarchy using the PCI tab of the Tools panel. This view is useful for systems with complex PCI bus configurations.

4.1.6 BIOS Parameters

VeB includes a screen that allows the engineer to easily view and edit the setting of BIOS parameters. BIOS parameters are analogous to the make flags used by previous AMIBIOS® releases. Each BIOS component includes an SDL file. SDL files define the parameters that are relevant to each BIOS component. Using VeB's BIOS parameter screen, it is not necessary to directly modify make files.



4.1.7 BIOS Build

A simple mouse click from within VeB builds the BIOS and displays any error messages that result from the build. Build output is displayed in VeB's lower pane. Double clicking on an error message in the window causes the related file to be opened in VeB's editor.

4.1.8 Context Sensitive Help

VeB includes a comprehensive help system for all chipset and OEM hooks. Pressing 'F1' while the cursor is located on the name of a hook function activates help for that function. Help includes a description of what the hook should do, who calls it, inputs and outputs, etc. Over time help may also include helpful hints or common pitfalls related to particular chipset or OEM hooks.

4.2 AMI Debug for AMIBIOS8

AMIBIOS8™ includes support for an integrated debugger. The AMIBIOS8 Debug Target returns information to the Host-side Debug Application over a selected interface (RS-232 & USB 2.0 "debug mode[4]"). The following sections describe the baseline features for these two components, and how they interact when in operation.

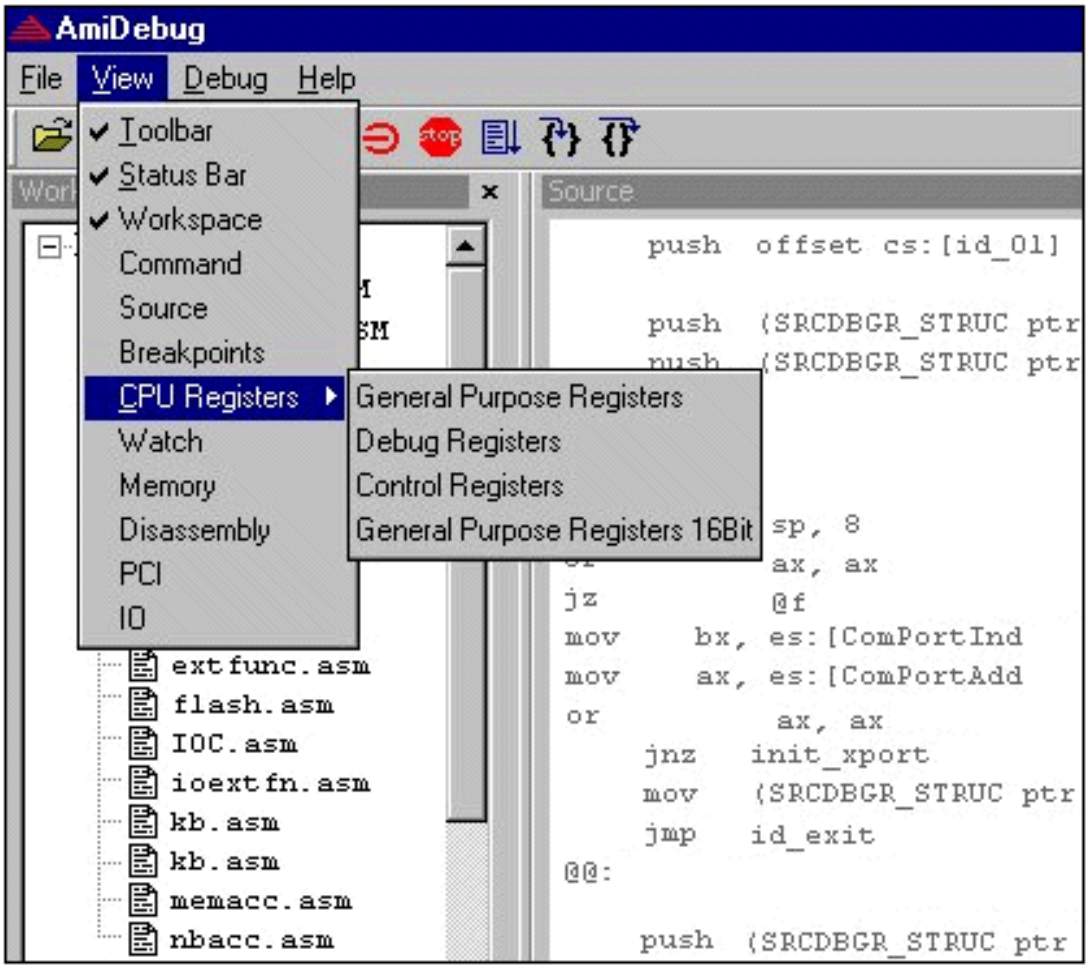
[4] The "debug mode" utilized by AMI Debug for USB target connections is described in Appendix C of the EHCI Controller Specification.

4.2.1 AMIBIOS8 Debug Target

To add debugging support to a AMIBIOS8™ BIOS the debugger eModule must be added to the BIOS project. Adding this component causes two binary modules to be placed into the ROM image. Together these BIOS resident debug components occupy less than 3k bytes of compressed space in the ROM. After it is initialized early in POST, the BIOS resident debug component waits for commands from the host system upon initialization. The host system can:

- Read & write CPU registers, including debug registers
- Read & write memory
- Read and write system IO
- Read and write PCI configuration space

The debug component also provides module load and move information so that the host may track which BIOS modules are resident in memory. The host system can also stop the execution of the BIOS at any time.



4.2.2 Host Debugger Program

The user interacts with the debug target though a Microsoft Windows application running on a host system. The host debugger program has similar functionality to an in circuit emulator (ICE), at a fraction of the cost. The host side application can be used to:

- Set and remove breakpoints
- View & modify memory/IO/PCI
- View & modify CPU Registers
- Disassemble & view source
- Load a file to targets memory
- View checkpoints
- Start & stop the execution of the BIOS, or trace through code
- Connect to debug target via RS-232 or USB Debug Port

4.3 Simplified Directory Structure

The directory structure in AMIBIOS8™ is organized by component to simplify project layout. An AMIBIOS8™ project is divided in four primary directories: CORE, CSP, BSP and BUILD.

The CORE directory includes object (OBJ) and include (INC) files that define the main functionality of the AMIBIOS®. This also includes the generic portion of any BIOS modules.

CSP stands for 'chipset support package'. This directory contains code pertaining to the system chipset (typically north bridge & south bridge). This also includes the chipset-specific portion of any BIOS modules.

BSP stands for 'board support package'. This directory contains code specific to a particular platform, including OEM customizations. This also includes the OEM-specific portion of any BIOS modules.

The BUILD directory is for temporary storage of files created during the BIOS build process.

Porting AMIBIOS8™ to a particular chipset involves porting routines in CSP directory. The files in the following **CSP** subdirectories are to be updated during chipset porting:

CSP	Directory containing the chipset specific files
CSP\COREMOD	Core files that require chipset specific modifications
CSP\INC	Chipset include files

Porting AMIBIOS8™ for a particular OEM involves porting routines in BSP directory. The files in the following **BSP** subdirectories are to be updated for OEM porting:

BSP	Directory containing the platform specific files (PCI IRQ Routing, I/O Initialization, custom silent boot logo, etc.)

BSP\COREMOD	Core files that require OEM specific modifications
BSP\CSPMOD	Chipset files that require OEM specific modifications
BSP\INC	OEM include files

Files in the following **BSP** subdirectories are not to be updated by OEM unless source override is required:

BSP\IO	Super I/O and System I/O files will go in this directory, if overridden.
BSP\FLASH	Flash support files
BSP\CPU	CPU support files
BSP\ADDONS	Extra binaries (CPU microcode patch, option ROM, etc.)

4.4 System Description Language (SDL)

Many AMIBIOS8™ components have configurable properties. For example, the BIOS size can be modified (256KB, 512KB, etc.). In AMIBIOS8™, all such properties are called "parameters." Parameters are described using System Description Language (SDL). An SDL file will accompany all BIOS components that have configurable parameters. VeB displays all parameters as described in Section 4.1.6.

All the flags - switches and variables – which were used in previous AMIBIOS® releases make process are now moved to System Description Language files under AMIBIOS8™. From these .SDL files, a new utility, AMISDL, generates a set of flags used in the make process (TOKEN.MAK) and a set of "MKF_…" equates to be used in source files (TOKEN.EQU).

4.5 Component Information Format (CIF) Files

All AMIBIOS8™ components can contain a variety of files:

- Source code (.ASM, .C, .INC)
- Pre-compiled objects (OBJ)
- Binary files (BIN, ROM)

Each component is released with a "packing list", using the Component Information Format (CIF). VeB uses this information to track the components in a particular BIOS project.

5 AMIBIOS8™ Usage Models

AMIBIOS8™ is a modular design, which makes it adaptable to any system board configuration. Some of the various usage models for AMIBIOS8™ are outlined below.

5.1 Initial Board Bring-Up

During the initial development phase of any project, a basic 'bring-up BIOS' is generated. The 'bring-up BIOS' is designed to only validate low-level functionality of the platform. Many BIOS features are disabled in this phase of development.

With a 'monolithic' BIOS model, extra work must be performed to deactivate high-level BIOS features, such as ACPI and SMBIOS. With AMIBIOS8™, the initial BIOS can be created without the need for this code in the project tree. After the initial development phase, the 'bring-up BIOS' can be transitioned to a full-featured product by adding features via eModules.

A 'bring-up BIOS' will typically utilize the following AMIBIOS8™ components:

- AMIBIOS8™ Core Components
- CPU Module, North Bridge & South Bridge Chipset Components
- Flash & Super I/O Modules
- Board Module, based on chipset reference platform
- AMIBIOS® Debugger

5.2 Embedded Platform

In many ways, BIOS for embedded platforms resembles a 'bring-up BIOS' for a desktop system. Embedded systems do not require many of the features used by a desktop system. However, embedded BIOS on current platforms require many of the management features used by server & workstation systems. AMIBIOS8™ features such as console redirection and remote flash are well suited for the embedded market, since many embedded systems do not utilize a VGA graphics interface.

An embedded platform BIOS will typically utilize the following AMIBIOS8™ components:

- AMIBIOS8™ Core Components
- CPU Module, North Bridge & South Bridge Chipset Components
- Flash & Super I/O Modules
- Board Module, based on initial board bring-up BIOS
- Console Redirection & Remote Flash for Headless Operation

5.3 Consumer Desktop

A desktop platform BIOS will typically utilize the following AMIBIOS8™ components:

- AMIBIOS8™ Core Components

- CPU Module, North Bridge & South Bridge Chipset Components
- Flash & Super I/O Modules
- Board Module, based on initial board bring-up BIOS
- ACPI Module
- SMBIOS Module
- Customized BIOS Setup Module
- Trusted Computing Group (TCG) eModule, for Trusted Platform Module (TPM) support
- USB Module, including support for bootable USB Mass Storage Devices

5.4 Notebook

A notebook platform BIOS will typically utilize the following AMIBIOS8™ components:

- AMIBIOS8™ Core Components
- CPU Module, North Bridge & South Bridge Chipset Components
- Flash & Super I/O Modules
- Board Module, based on initial board bring-up BIOS
- ACPI Module, with support for ACPI-compliant Embedded Controller
- SMBIOS Module
- Support for PCMCIA/Cardbus Controller
- USB Module, including support for bootable USB Mass Storage Devices
- HotKey eModule, with support for OEM-specific keyboard shortcuts
- Extended power management, such as Intel SpeedStep™ Technology
- Embedded Controller (EC), including related ACPI objects

5.5 Server

A server platform BIOS will typically utilize the following AMIBIOS8™ components:

- AMIBIOS8™ Core Components
- CPU Module, North Bridge & South Bridge Chipset Components
- Flash & Super I/O Modules
- Board Module, based on initial board bring-up BIOS
- ACPI Module
- SMBIOS Module, including error logging to GPNV
- Microsoft Windows Hardware Error Architecture (WHEA)
- Console Redirection & Remote Flash for Headless Operation
- IPMI eModule