



PhoenixBIOS 4.0 Release 6.1

Feature Descriptions

24 March 2001

Copyright

PhoenixBIOS 4.0 Release 6.1 Feature Descriptions

Copyright © 2001, Phoenix Technologies Ltd. All Rights Reserved.

Printed in U.S.A. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of Phoenix Technologies Ltd.

Disclaimer

Phoenix Technologies Ltd. makes no representations or warranties with respect to the design and documentation herein described and especially disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Phoenix Technologies Ltd. reserves the right to revise this design and associated documentation and to make changes from time to time in the content without obligation of Phoenix Technologies Ltd. to notify any person of such revisions or changes.

Comments

Please send all comments and suggestions regarding this manual to:

Technical Publications
Platform Enabling Division
Phoenix Technologies Ltd.
135 Technology Drive
Irvine, CA 92618

Contents

Feature Descriptions for PhoenixBIOS 4.0 Release 6.1	1
I. PhoenixBIOS Platform Support	1
A. Phoenix Silicon Support	1
B. Phoenix Multiple Configuration	2
C. Phoenix P6 Update	2
D. Phoenix NuKBC	2
E. Phoenix BootBlock 002	2
F. Motherboard Configurable Devices (MCD)	4
G. Phoenix Plug-and-Play 4.3	4
H. Phoenix Intelligent I/O (I ₂ O)	5
II. PhoenixBIOS Memory Support	6
A. DRAM	6
B. SRAM	6
C. Cache	6
D. CMOS	7
E. Option ROMs	7
F. Shadow Memory	7
G. Conventional Memory	7
H. Upper Memory	7
I. High Memory	7
J. Extended Memory	7
K. Flash ROM	7
III. PhoenixBIOS Internal Bus Support	8
A. The ISA Bus	8
B. The EISA Bus	8
C. The PCI Bus	8
D. The SMBUS	9
E. VESA Universal Memory Architecture (VUMA)	9
IV. PhoenixBIOS External Bus Support	9
A. Serial Ports	9
B. Parallel Ports	9
C. SCSI	9
D. USB	10
E. IEEE 1394	10
F. Legacy Free	11
V. PhoenixBIOS Mass-Storage Support	11
A. Floppy Drives	12
B. Fixed-Disk Drives	12
C. PC Cards (PCMCIA)	14
VI. PhoenixBIOS Boot Features	16
A. Phoenix MultiBoot III™	16
B. Phoenix QuietBoot	16
C. Phoenix QuickBoot	17
D. Setup Boot-Menu Features	17
VII. PhoenixBIOS POST Features	17
A. POST Services	17
B. The NVRAM Manager	17
C. POST Configuration Features	18
VIII. PhoenixBIOS Setup Features	19
IX. PhoenixBIOS Security Features	20
A. Setup Security-Menu Features	20
B. BIOS Security Services	20
X. PhoenixBIOS Power-Management Support	20
A. Phoenix Miser 4.0	20
B. Phoenix APM	21
C. Phoenix ACPI BIOS	21
D. Phoenix Save-To-Disk	22
XI. PhoenixBIOS System-Management Features	22
A. Phoenix SMBIOS 2.3.1	22
B. Phoenix IPMI	22
C. Phoenix Universal Console Redirect (UCR)	23
D. Phoenix Wired for Management (WfM)	23
XII. PhoenixBIOS Build Features and Utilities	24
A. PhoenixBIOS ROM Maker Tools	24
B. PhoenixBIOS Deployment Utilities	24
Index	27

Feature Descriptions for PhoenixBIOS 4.0 Release 6.1

The following features of PhoenixBIOS 4.0 Release 6.1 provide software engineers the best solutions for adapting platformware to a new platform.

The following describes for PC manufacturers and engineers the standard features included with PhoenixBIOS, Phoenix ServerBIOS, and Phoenix NoteBIOS, and special features purchased separately. See your Phoenix Sales Representative for details.

See the *PhoenixBIOS 4.0 User's Manual* (on the Phoenix Web site) for descriptions of:

- Setup
- BIOS messages
- POST tasks and beep codes
- Run-Time Services
- Interrupt Vector Table
- BIOS Data Area (BDA)
- Extended BIOS Data Area (EBDA)

I. PhoenixBIOS Platform Support

The following Platform-Management features handle the first tasks faced by an engineer in implementing PhoenixBIOS on a new platform. They address the fundamental tasks posed by the following platform-design issues:

- The CPU.
- The chipset.
- The keyboard controller.
- The video controller.
- The boot block.
- IRQ Routing.
- Intelligent I/O (I²O)
- Accelerated Graphics Port (AGP)

Addressing these tasks are the following PhoenixBIOS features:

1. Phoenix Silicon Support
2. Phoenix Multiple Configuration
3. Phoenix P6 Update
4. Phoenix NuKBC
5. Phoenix BootBlock
6. Phoenix Motherboard Configuration Devices (MCD)
7. Phoenix PnP 4.3
8. Phoenix I²O

The following describes each of these platform-management features.

A. Phoenix Silicon Support

Phoenix provides silicon support for the widest range of these devices:

- CPUs
- Chipsets
- Super I/O devices
- Audio Controllers
- Video Controllers
- LAN Controllers

Contact Phoenix Silicon Support for more information.

B. Phoenix Multiple Configuration

Phoenix Multiple Configuration provides BIOS source-code customers with an easy method to automatically configure any one of a range of devices. Silicon devices such as chipsets, CPUs, and super I/Os often go through several updates, called revisions. CPU type and speed can be product options or even upgrades. Each BIOS shipped within the product life cycle should be able to detect with revision or upgrade is present, and make adjustments (usually to the chipset) to maintain conformity and enhance system performance. For example, when a PC-user upgrades the CPU, the BIOS should detect the change and make adjustments in the chipset or other devices to make best use of the upgraded product.

Phoenix Multiple Configuration initializes device registers with values based on the condition or state of another device, such as the type and speed of the CPU. It also enables BIOS upgrades to work successfully when installed with older versions of the product.

C. Phoenix P6 Update

The Intel **Pentium Pro** and later P6 class processors have the capacity to correct specific errata through the loading of an Intel-supplied data block, called the **BIOS Update**, to the processor itself during POST.

Each BIOS Update is tailored for a particular version of the **Pentium Pro**. Intel considers the combination of a particular revision of the processor and a BIOS update as the equivalent of a processor stepping. The Update is an encrypted binary with a header but no executable code. Each BIOS Update must match the specific family, model, and the stepping of the processor as returned by the CPUID instruction or else the processor rejects it.

The **Phoenix P6Update** is the PhoenixBIOS 4.0 implementation of this process. It consists of the **Update Loader** that loads the Update from the flash part to the CPU during POST, and an INT 15 interface to replace Updates already in the BIOS with new ones.

D. Phoenix NuKBC

NuKBC (Keyboard Controller) supports all known PS/2 Devices and all known keyboard technologies (including ACPI Embedded Controllers and USB Device Emulation).

NuKBC includes the following functionality:

- Initialization and configuration of the ACPI Embedded Controller.
- Correct configuration of Keyboard Support with USB Device Emulation.
- Support and detection of Intellimouse.
- Support of keyboard and mouse port-swapping.
- Independent operation of the device interrupt code and the device service code.
- Improved keyboard-and-mouse interrupt handlers.
- Improved keyboard and mouse device initialization and configuration.
- Improved password support, which includes dual 16-byte passwords.
- Independent operation of features (not dependent upon a particular keyboard controller).
- Interface with existing calls to the keyboard component.
- Replacement modules to facilitate System BIOS Debugging.
- Inclusion of INT16h Extensions to test NUKBC configuration and performance.

For the standalone firmware required by specific keyboard controllers, see the MultiKey products.

The Setup Menu Keyboard features include these user-selectable options:

- **NumLock** Power-On State.
- **Typematic** Rate and Delay.
- **Audible Key click**.

E. Phoenix BootBlock 002

The **Flash ROMs** used in many systems today offer engineers and end users the advantage of electronically reprogramming the BIOS from a DOS environment without physically replacing the BIOS ROM. This advantage, however, does create a possible hazard: power failures or fluctuations that occur during updating the Flash ROM can damage the BIOS code, making the system unbootable.

To prevent this possible hazard, many Flash ROMs include a special non-volatile region, which can never be erased. This region, called the **BootBlock**, contains a fail-safe recovery routine. If the BootBlock finds

corrupted BIOS (checksum of system BIOS fails), it boots into the crisis recovery mode and loads a BIOS image from a special crisis diskette and flashes the BIOS binary image into the Flash ROM that replaces the corrupted BIOS with an uncorrupted one.

Phoenix BootBlock supports two recovery routines, MINIDOS.600 and FULLDOS.600, allowing OEMs to use their own Crisis Recovery Disk and recovery routines if they choose.

The following are the features of BootBlock 002:

1. **Dynamic Size Generation**

Previously, the BootBlock was limited to 8KByte. The BootBlock 002 supports BootBlock size larger than 8Kbyte while maintaining the BIOS compatibility. The BootBlock code size is limited only by the physical size of the BootBlock area in the Flash device.

2. **Compression**

Book Block 002 now supports only one type of compression, LZSS, the most efficient and only compression method provided in BootBlock. You can select which modules to compress.

3. **POST Dispatch Manager**

The **BootBlock PDM** manages the BootBlock modules. The BootBlock PDM supports these main services:

- Find module in ROM
- Get module size
- Expand module to address

4. **SMM Secured Flash**

The BootBlock SMM uses **System Management Mode** to provide secured Flash process.

BootBlock SMM of two main pieces:

- The SMM initialization
- The SMM secured flash dispatcher

In order for the SMM secured to work, the engineer must provide the secured recovery code to perform the flashing execution (this code must reside in the SMM memory area) and the application code to trigger the SMI to enter into the SMM mode and executing the flash code.

5. **Multiple Processors**

The BootBlock handles Multiple Processors' platforms and enables the primary processor to execute the crisis code.

6. **CPU Type**

Systems with newer CPU than Pentium P54 require some initialization. The BootBlock supports this requirement.

7. **Memory Auto sizing and Auto typing**

The BootBlock supports memory auto sizing and memory auto typing for the Fast Page Memory type, EDO type and SDRAM type. The BootBlock detects the type of memory used by the platform, the size of this memory and set the memory controller chipset accordingly.

8. **Video VGA**

In Crisis mode, when this feature is enabled the Flash process is displayed on a VGA monitor.

9. **A16 Swapping**

The BootBlock supports A16 swapping. It enters to crisis mode when A16 is switched to crisis recovery.

10. **Boot Devices Support**

The BootBlock supports executing the crisis recovery program from the Legacy floppy, USB (UHCI only) or the LS 120 media (ATAPI removable disk).

F. Motherboard Configurable Devices (MCD)

Phoenix MCD (Motherboard Configurable Devices) is the name of the PhoenixBIOS support of all the configurable I/O devices on the motherboard. Such devices include the physical device, such as a **Super I/O** and the logical devices such as COM, LPT, Floppy, and IDE (which can be part of the Super I/O).

The Plug-and-Play specification makes the BIOS responsible for configuring motherboard I/O devices as well as PnP ISA and PCI devices to non-conflicting states.

The problem is to bring the diverse range of motherboard devices under PnP BIOS control such that their resource configuration is optimized for overall system operation. Optimization entails:

- The prevention of a system hang, due to conflicting resources assigned to devices.
- Prioritizing devices and their resources for maximizing the number of system devices enabled.
- Setup detection and notification of user-created resource conflicts

In addition to these requirements, the guidelines include a set of rules for enabling OS configuration of motherboard devices. While OS configuration is a Microsoft requirement, secured settings are a customer requirement. Phoenix MCD fills both of these requirements.

MCD also supports SoundBlaster Pro, AdLib FM and Codec features incorporated into a chip, dependent functions for the IO/IRQ/DMA resources available for legacy DOS support, and dependent functions required for PC 2000 compliance.

Installed MIDI functionality supports MPU-401 serial port interfacing with an external MIDI device, independent or dependent functions available from this chip to add legacy PnP support, and independent and dependent functions required for Microsoft compliance.

A Power-Management Service Routine (PMSR) for Motherboard Configurable Devices provides generic power-management code for all MCD devices, (e.g., COM, LPT, Floppy, and IDE), eliminating the need for a separate PMSR for every device. This generic PMSR reduces code size and makes customizing the code easier.

G. Phoenix Plug-and-Play 4.3.

For operation, most devices connected to PCs require the exclusive use of assigned elements of **system resources** which include:

- Direct Memory Access (DMA)
- Interrupt Request Lines (IRQs)
- I/O Address space
- Memory Address space (RAM and ROM)

For early PCs with only Legacy devices on an ISA (Industry Standard Architecture) bus, configuring new devices required selecting options in the Setup menus and physically changing jumpers on the devices before installing. The resulting conflicts in the assignment of the same resources to different devices often caused boot and system failures.

PCI, EISA, and ISA Plug-and-Play devices were designed to address this problem. Communicating with one of these new devices, PnP software (e.g., Phoenix PnP-4.x or Windows 95 and later) performs the following procedures:

1. Obtains a list of preferred and acceptable resource allocations from the device.
2. Selects and allocates (reserves) resources for the device.
3. Configures the device (tells it which resources to use) by setting the device's configuration registers.

Phoenix PnP 4.3 supports these two basic BIOS functions:

- **Dynamic System Configuration** detects and configures PCI, PnP ISA, and embedded ISA devices. See PhoenixBIOS POST Features below.
- **Runtime Services.** As defined in the *Plug and Play BIOS Specification*.

Phoenix PnP 4.3 includes the following standard core features:

- **Low-to-High Resource Allocation.** Supports the bottom-up resource allocation required by Multiple Host buses of Intel's NX chipset.

- **New NVRAM Interface.** Allows the BIOS and external applications to save data to the NVS parameter block in flash or other storage medium using data signature.
- **Support for 256 PIRQs.** Increases the number of PIRQs from 32 to 256.
- **Claim unused shadowed RAM in device nodes.** A new device node representing the location and amount of unused shadow memory prevents the OS from placing memory-mapped I/O for PnP ISA/PCI in these shadowed regions.
- **Optional Expanded I/O Resources for PCI Devices.** Allows PCI device I/O assignments to be placed in legacy ISA aliased I/O addresses. Therefore, more I/O addresses can be allocated to PCI devices.
- **Print of the screen from Setup.** If the parallel port is left disabled in Setup to enable configuration by the OS, the user can print the Setup screen.
- **Legacy Device Detection Part of Setup CDR.** In Setup, the Conflict Detection and Resolution (CDR) program takes into account devices found in POST, such as legacy COM, LPT, floppy, or IDE devices.

Phoenix PnP 4.3 also includes the following **optional installable** features:

- **Automatic caching of PCI expansion ROM's.** Optional support lowers initialization time and increases performance.
- **PCI-IRQ Interface.** External applications can now save a PCI device IRQ and latency information using the NVRAM API. The application stores the setting for any card to be configured into NVRAM. During the next boot, the BIOS reads the configuration information and configures the PCI card accordingly.
- **Smart PCI-IRQ Allocation.** Internally allocates resources for devices that are left disabled in POST for the OS to configure. When MCD devices and non-IPL PnP ISA cards are left disabled, the OS can allocate the necessary IRQs to both PCI and non-PCI devices. See PCI-IRQ Routing below in "PhoenixBIOS POST Features."
- **PCI Audio.** This feature allocates IRQ 5 to PCI audio cards whenever IRQ 5 is available. DOS applications and sound cards require the use of IRQ 5.
- **Setup Selection of Boot VGA Device.** Either AGP or PCI can be selected as a boot VGA device. This feature supports AGP whenever a PCI VGA device has been plugged into the system for dual monitor support.
- **Integrated Subsystem ID Support.** Provides generic handling of IDs of motherboard PCI devices to comply with Windows 95 Certification.

H. Phoenix Intelligent I/O (I₂O)

For some time, mainframes have used special I/O processors that provide a level of abstraction between the CPU and I/O devices. Although the CPUs in today's PCs are very fast, they need as much help with I/O processing as the mainframes. The I/O interrupts from SCSI drives and LAN cards, for instance, can interrupt the CPU many times during a single operation, greatly reducing performance.

The new I₂O technology, developed by members of the I₂O SIG, uses an **I/O Platform (IOP)**, a node inside a PC consisting of:

- An I₂O processor like the Intel i960.
- An I/O ROM for storing the I₂O Run-Time Operating System (IRTOS) and device drivers.
- I/O DRAM memory.
- The I₂O-compatible devices controlled by the I/O processor such as SCSI and IDE hard drives, LAN cards, CD ROMs, and RAID subsystems. I₂O block-oriented storage devices are also called **block-storage devices**.

When the CPU sends a data request to the IOP, the I/O processor handles most of the interrupts required for the data transfer and then sends the data as one large block to the CPU. This new off-loading of interrupt tasks from the CPU greatly boosts performance.

With an intelligent device between the peripheral and the operating system, peripheral vendors can write one standard driver for each type of peripheral that will work with any operating system, making I/O cards simpler to implement.

The I/O processor is a programmable device run by an IRTOS. The processor communicates with the CPU and other I₂O devices through **Messages**, data structures that contain a header and a data payload. The I/O processor also loads and runs the **Device Driver Modules (DDMs)**, the drivers of I₂O devices.

Each block-storage device is identified with a **Target ID (TID)**, the logical address of the device.

Each IOP has a descriptor. A descriptor contains the following information: IOP memory address, an **ioHandle** (for code identification), flags, and data structure pointers. The IOP descriptors are stored in a linked list. During POST, the descriptors live in PMM memory, but after INT 19h, they are moved to the EBDA.

PhoenixBIOS support for I₂O boot consists of these pieces:

- **I₂O Initialization.** Initialize I2O global variables during POST Task 47h (ioInitJ).
- **PCI Bus Enumeration** – Find all IOPs and initialize them during POST Task 49h (pciInitJ).
- **I₂O Messaging** – This is the BIOS code that actually communicates with an IOP. All communication is done through I2O Messages.
- **INT 13h Functions** – Convert INT 13h drive accesses to I2O block-storage-device accesses.
- **MultiBoot II Support** – Add I2O block-storage devices to the MultiBoot II Setup menus.
- **I₂O Setup Menu** – For technical configuration of I₂O operations (See the *PhoenixBIOS 4.0 User's Guide*, "The I₂O Menu").

The BIOS I₂O Messaging code needs a buffer to receive messages from the IOPs. During POST, the buffer lives in PMM memory. After INT 19H, the buffer is moved to the EBDA.

II. PhoenixBIOS Memory Support

During the Power On Self Test (POST), the BIOS can initialize and test memory, which falls into two categories, **electronic memory** and **mass storage**. Mass storage includes floppy, hard disk, CD-ROM, and high-capacity floppies such as Iomega Zip drives. Electronic memory includes DRAM, SRAM, CMOS, Flash, and ROM. Here is a more thorough explanation of the different types of memory.

A. DRAM

Dynamic Random Access Memory, also known as DRAM, is a type of memory for temporarily storing information currently being used by the system. It is like a worksheet for temporary storage and execution of programs. The contents of DRAM are lost when the power to the system is turned off. DRAM contains its information by storing electrical charges inside many capacitors. All capacitors, however, lose their charge over time and the information they are supposed to store. In order to keep this from happening the capacitors are continuously reading and rewriting the information in a process known as refreshing. DRAM SIMMs or DIMMs are typically used for main memory storage. The advantage of DRAM is that it is relatively inexpensive.

B. SRAM

Static Random Access Memory, also known as SRAM, is in many ways the same as DRAM except for the way the information is stored. For SRAM, the information in memory is stored in flip-flops (an electronic circuit which is like a switch, the circuit is either on (1) or off (0)). Flip-flops do not have to be refreshed, unlike the capacitors that make up the DRAM. Therefore, SRAM is faster than DRAM, however, it costs more. SRAM will also lose its contents once the power is turned off to the system. SRAM is usually used in situations where speed is very critical.

C. Cache

Cache can be thought of as being an in between fast CPU and a slow memory. The cache holds the most current program(s), which the user is working on. If the user needs some information, the system first checks the cache. If the information is not in the cache, the memory (RAM) is checked. If the information is not in RAM, then the needed information is brought from the hard drives or floppy diskette. Cache can help speed up the system by cutting down on accessing the RAM or hard drive. Many computers have two types of cache, which run conjointly to greatly speed up processing. The first is part of the CPU. The second is physically located outside the CPU.

D. CMOS

Complementary Metal Oxide Semiconductor also known as CMOS is another type of memory. Selections made in Setup—such as the type and size of a floppy drive or hard drive information—are stored in CMOS. CMOS also contains a Real Time Clock (RTC), which is responsible for keeping the current time. The main advantage of CMOS memory is that its contents are not lost when the power to the system is shut off. Once the systems power is off, a battery located on the motherboard powers the CMOS and the RTC. The battery supplies enough power to keep the RTC active and to retain the CMOS memory contents.

E. Option ROMs

ROM is a special type of memory that can only be read; it cannot be altered or written to. ROM also retains contents when the computer is turned off. When people think of ROM they normally associate it with the system BIOS. Option ROM's refer to the software boot code that usually resides on external cards, such as PnP- ISA, PCI, and ISA plug-in cards. This code has the responsibility for the card's initialization and operation.

F. Shadow Memory

During early POST, the BIOS "shadows" (copies) itself from the slow 8-bit ROM BIOS contents into faster 32 or 64-bit RAM, dramatically speeding up the execution of POST tasks. The initial ROM address region overlies the new RAM area. This causes ROM accesses to be redirected into the faster operating RAM. After the information is copied from ROM to RAM, it is write-protected.

G. Conventional Memory

Conventional memory consists of the first 640 kB of DRAM memory. Conventional Memory is critical because DOS and DOS-based programs run within this memory space. The Interrupt Vector Table and BIOS data area also reside in conventional memory.

H. Upper Memory

Upper memory consists of the 384 kB of memory area above conventional memory. This upper memory contains the video RAM, video BIOS, system BIOS, and possibly option ROMs from PCI or PnP add in cards. The remaining free space within the upper memory area is called upper memory blacks. The upper memory blocks can be used for running device drivers, memory-resident programs, or for storing small programs.

I. High Memory

The high memory is the first 64K of extended memory. MS-DOS 5.0 and later operating systems load much of its operating code into high memory instead of conventional memory to make more of conventional memory to be available to other applications.

J. Extended Memory

Extended memory is all the system memory beyond 1 MB. Extended memory is on all computers that have at least a 286 CPU or higher. Older operating systems required an extended-memory manager, such as HIMEM.SYS. Windows and Windows based applications require extended memory.

K. Flash ROM

In most PCs, the BIOS are stored on Flash ROMs, which can be electrically rewritten and reprogrammed. During early POST, the BIOS copy itself from the Flash ROM into DRAM memory. Engineers and end users can update the BIOS with special Flash utilities such as Phoenix WinPhlash or Phlash16.

III. PhoenixBIOS Internal Bus Support

PhoenixBIOS supports the following standard buses and the bridges that connect them (PCI-ISA and PCI-EISA).

A. The ISA Bus

IBM developed the **Industrial Standard Architecture (ISA)** bus as an extension to the XT bus for its 80286-based PC/AT system. It is sometimes referred to as the *AT bus* or *ISA expansion bus*. It adds the following features to the XT specification:

- Eight additional data lines that allow data transfers in 16-bit blocks
- Four additional address lines that allow a 16-megabyte address space
- Backward compatibility with the XT specification
- Three additional 16-bit DMA channels
- Support for alternate bus masters

The physical ISA expansion connector adds a second set of 36 pins to the 62-pin XT connector. These extra pins are separated from the XT pins by a gap of about a quarter of an inch, so that an 8-bit XT expansion board will fit into an ISA slot and still function properly.

During POST, PhoenixBIOS configures both Legacy (non Plug-and-Play) ISA and Plug-and-Play ISA devices, using configuration data stored in NVRAM.

B. The EISA Bus

The **Expanded Industrial Standard Architecture (EISA)** bus supported by PhoenixBIOS adds several important features to the ISA specification:

- Sixteen additional data lines that allow data transfers in 32-bit blocks
- Eight additional address lines that allow a 4 gigabyte address space
- Backward compatibility with the XT and ISA specifications
- Three new types of DMA transfers, including a 32-bit burst mode
- Level-sensitive interrupts
- Sophisticated bus control arbitration

The physical EISA expansion connector adds 55 signals, on 90 new pins, to the ISA connector by incorporating a two-tier design. EISA slots will accept ISA or EISA expansion boards. EISA boards will fit into the lower tier of new connections, ISA boards will not.

C. The PCI Bus

PCI, an acronym for **Peripheral Component Interconnect**, is a local-bus architecture developed by Intel and introduced in 1993. PCI is a bus with much higher throughput than the standard ISA (Industry Standard Architecture) bus used in PC-type computers for years. Most of today's computers use a PCI bus. The PCI bus is a local bus, that is, it carries data directly from the expansion slots, where system devices are added, to the CPU. Data is typically transferred in groups of 32 or 64 bits.

The PCI Bus Specification, which describes the PCI BIOS software interface functions, offers a hardware-independent way to support PCI devices. The PCI bus was one of the first systems to popularize the use of Plug and Play devices.

For every PCI device available, the BIOS:

- Identifies the physical device
- Configures the card according to the settings provided
- Maintains PCI device information

Phoenix BIOS, which is fully PCI 2.1 compliant, supports the following PCI features:

- Any PCI 2.1 compliant bus configuration of any combination of width and depth
- Setup options that can be set to exclude IRQ, UMB, or I/O resources

- Setup options to shadow and initialize expansion ROMs
- Setup support of Latency Timer
- Setup support of Plug and Play Operating Systems
- ESCD or bitmap resource storage
- PCI expansion ROMs cached
- PCI audio support
- IRQ sharing that's spread across available IRQs

D. The SMBUS

Intel's **System Management Bus** is a two-wire interface used for communication with intelligent devices, such as a Smart Battery. This interface builds on the defined I2C specification for support of serial communication between simple logic devices.

E. VESA Universal Memory Architecture (VUMA)

VUMA is a specification that gives VUMA devices direct access to memory. Using the VUMA System BIOS Extensions (SBE), the video BIOS and other GUI-specific software can control the VUMA hardware without specific knowledge or direct hardware access. Using the VUMA Video BIOS Extensions (VBE) will allow the system BIOS and other GUI specific software to access the VUMA hardware without specific knowledge or direct hardware access.

VUMA support gives system-memory access to non-system controller devices. These devices, called VUMA devices have their own memory controller and access system memory directly. VUMA devices can potentially access all of system memory.

IV. PhoenixBIOS External Bus Support

PhoenixBIOS supports the following communication architectures:

A. Serial Ports

PhoenixBIOS expects the system to include a National Semiconductor 8250 serial port controller or equivalent logic. PhoenixBIOS supports up to four serial ports, which are accessed through address ports 02F8h-02FFh and 03F8h-03FFh.

The NS 8250 handles data with the following characteristics:

- 5, 6, 7, or 8 bits
- 1, 1.5, or 2 stop bits
- Even, odd, or no parity

The serial port controller can operate at speeds of from 110 to 19,200 bps.

B. Parallel Ports

PhoenixBIOS supports parallel ports that transfer 8 bits of data at standard TTL levels. The parallel ports can be called port 1, 2, or 3. Some adapters may provide a bi-directional data port. PhoenixBIOS supports up to four parallel ports.

The parallel ports are accessed through address ports 0278h-027Ah, 0378h-037Ah, and 03BCh-03BEh.

Enhanced Parallel Port (EPP) and **Extended Capabilities Port (ECP)** are bi-directional parallel-port technologies frequently used to support network adapters.

C. SCSI

The Small Computer System Interface is a popular communication architecture often used for fixed-disks and CD-ROM drives.

D. USB

The **Universal Serial Bus (USB)** is a medium-speed bus extension to the PC architecture that focuses on Computer Telephony Integration (CTI) and consumer and productivity applications.

The USB specification defines the following requirements for the Universal Serial Bus:

- Ease-of-use for PC peripheral expansion.
- Low-cost solution that supports a transfer rate of 12 Mbit/sec.
- Full support for the real-time data for voice, audio, and compressed video.
- Protocol flexibility for mixed-mode isochronous data transfers and asynchronous messaging.
- Integration in commodity-device technology.
- Comprehend various PC configurations and form factors.
- Provide a standard interface capable of quick diffusion into product.
- Enable new classes of devices that augment the PC's capability such as Hot Attach and Detach.

PhoenixBIOS USB is the PhoenixBIOS support for:

- USB Keyboard and Mouse
- USB Floppy (with hot swapping)
- USB Hard Disk
- USB CD ROM

It also provides USB Legacy support for Port 60/64 I/O trapping for keyboard and mouse.

Because the Phoenix USB supports existing BIOS services, it requires no new external interface and is implemented in the core.

The USB architecture of PhoenixBIOS provides these functions:

- USB Services
- The Legacy Interface
- Host Controller Services

PhoenixBIOS USB Services provide:

- POST detection and registration of the USB Host Controller and the enumeration of each device on the USB bus. Enumeration may be performed again at run time.
- POST initialization of the Legacy Interface.
- Run-Time dispatching of high-level commands to the Host Controller Services.
- During Run Time, the PnP-PCI component uses named entry points to access USB Services. The Legacy interface and Host-Controller activity use SMIs or IRQs.

The **Legacy Interface** provides:

- Run-time trapping and emulation of I/O accesses by the OS and application programs.
- Run-time generation of IRQs or SMIs to access USB Services.

Host Controller Services manage communications with the USB devices through these separate modules in the NUBIOS\BUS\USB component:

- The **Universal Host Controller Interface (UHCI)** sponsored by Intel or
- The **Open Host Controller Interface (OHCI)** sponsored by Microsoft, Compaq, and others.

E. IEEE 1394

The new, very fast 1394 external-bus standard supports data-transfer rates of up to 400 MBPS (400 million bits per second). Products supporting the 1394 standard go under different names, depending on the company. For example, Apple, which originally developed the technology, uses the trademarked name **FireWire™**.

A single 1394 port can be used to connect up to 63 external devices. In addition to its high speed, 1394 also supports isochronous data, which is data delivered within certain time constraints and at a guaranteed

rate. This makes 1394 ideal for devices that need to transfer high levels of data in real-time, such as video devices.

PhoenixBIOS 1394 supports booting from a device connected by a 1394 bus such as a 1394 hard drive, CD-ROM, or High Capacity Removable Media (HCRM). After normal IDE devices are autotyped, the 1394 code scans the 1394 bus for devices that it can control. Once 1394 builds the list of these devices, it attempts to autotype them and then picks an INT 13h protocol to control them.

PhoenixBIOS support for 1394 has special provisions for these two types of 1394 support:

- **OHCI** — **Open Host Controller Interface** (There is a 1394 OHCI spec as well as a USB OHCI spec.)
- **PELE** — 1394 Host Controller silicon designed and built by Apple and SymbIOS.

F. Legacy Free

The market trend toward less costly and more accessible PCs is driving OEM demand removing obsolete, slow, complicated, and often poorly understood interfaces. Phoenix Legacy-Free supports the two industry initiatives, the "Legacy-Free PC" and the more global "Easy PC," both of which will dramatically change the hardware, firmware, and Operating System of servers, workstations, desktops, and notebooks.

They start with the removal of some legacy interfaces that negate the progress of newer technologies. The initiatives include reducing the unique connectors (serial, parallel, keyboard, and mouse) on the back of these systems. They also include the removal of "legacy I/O" (which is interchangeable with ISA architecture), ISA slots, ISA devices, standard legacy PC features such as COM and LPT ports, keyboard controller (KBC), and the core architecture related to bus support, legacy-device interrupts and other resources. The ISA regime is to be replaced with those governing USB and IEEE 1394.

The basic goal of these requirements is that the operating system, devices, and end users cannot detect the presence of the following: ISA bus, slots or devices; legacy floppy disk controller (FDC); and PS/2, serial, parallel, and game ports.

There are several exceptions to the above requirements, including:

- COM ports may be used (usually at non-standard addresses) as debug ports, but must be reported in the debug port table under ACPI (See below).
- Mobile (Notebook) PCs may use KBC decoded at 60h, 64h

Legacy reduced platforms mean these platforms still have a SIO and/or KBC but they still behave like Legacy Free. Most of the time, the terms "legacy free" and "legacy reduced" are used interchangeably.

As a result, the main elements of a legacy free/reduced project entail elements that can be separated into primary and secondary Requirements.

V. PhoenixBIOS Mass-Storage Support

The BIOS may be required to boot an operating system from any one of the following mass-storage devices:

- Floppy Drives
- IDE and SCSI fixed disks
- ATAPI CD-ROMs
- Network cards (remote boot)
- PC Cards (PCMCIA)
- USB storage media (floppy, hard disk, and CD ROM)
- Iomega Zip and other high-capacity floppies
- IEEE 1394 (FireWire) storage media

The following describes PhoenixBIOS support for these devices.

A. Floppy Drives

PhoenixBIOS supports these standard floppy drives

- 360 kB 5.25"
- 1.2 MB 5.25"
- 720 kB 3.5"
- 1.44 MB 3.5"
- 2.88 MB 3.5"
- 1.44 MB 3.5" USB

PhoenixBIOS also supports:

- **USB Floppy boot** with hot swapping
- **3-Mode Floppy**, an installable feature that meets the requirement of some Japanese computers that enables a 1.44 MB floppy drive to read and write a 1.2 MB format.

B Fixed-Disk Drives

The following lists the OEM-installable PhoenixBIOS IDE fixed-disk features.

1. LBA

LBA (Logical Block Access) replaced **CHS** (Cylinder/Head/Sector) as the transfer mode. A feature of INT 13 Extensions, it requires 1 bit of CMOS per drive. All drives currently support CHS; some larger drives support LBA in addition to CHS. The BIOS switches to CHS if the drive does not support LBA.

2. INT13Extensions

The **INT 13 Extensions** differ from traditional fixed-disk INT 13 routines by using Logical Blocks Addressing (LBA) in place of Cylinder-Head-Sector (CHS) to address the hard disk. In the event that a drive does not support LBA, the BIOS switches to CHS addressing.

For a description of the INT 13 Extension APIs, see:

PhoenixBIOS 4.0 User's Manual, available on the Phoenix Web site at:
<http://www.phoenix.com/pcuser/userman.pdf>

BIOS Enhanced Disk Drive Specification Version 1.1, available at the Phoenix Web site at:
<http://www.phoenix.com/products/specs-edd11.pdf>

3. LBASSIST

Hard drives larger than 8.4GB require either LBA assisted (LBASSIST) or Phoenix Bit Shift (PBS) described below. They are mutually incompatible. Most OEMs and test labs prefer LBASSIST, which is the industry standard. LBASSIST does not work with drives that do not support LBA. For a further discussion of both LBASSIST and PBS see the document "*BIOS Enhanced Disk Drive Specification, Version 1.1*" located on the Phoenix Web site at <http://www.phoenix.com/products/specs-edd11.pdf>

4. Phoenix Bit-Shift (PBS)

If your system is over 8.4 GB and does not require LBASSIST, PBS has slightly less space requirements than LBASSIST described above. This feature does not work with drives that do not support LBA.

5. ModelTypeDisplay

An OEM-selectable message that displays the model type of the hard disk and CD ROM during POST and Setup. Automatically installs POST Auto.

6. Model Type

An OEM-selectable message that displays the model type of the hard disk and CD ROM during POST but not during Setup. Requires POSTAuto.

7. 4Drive

The 4Drive option changes the maximum number of drives from 2 to 4 and calls in a small amount of code for shadowing the additional drive-mapping information.

8. 32BitIO

32BitIO supports 32-bit transfers on Local Bus and PCI drives.

9. BlockPIO

Block Programmed Input/Output transfers data in blocks of 2, 4, 8, or 16 sectors, as specified by the end user in Setup.

This feature requires 11 bits of CMOS per drive. It calls a small amount of code to shadow the block PIO bits as well as two routines for supporting block PIO. Some drives return a very large number when they are autotyped for multiple-sectored block size. Such a number can easily cut drive performance in half. For this reason, the user-entry limit is still 16 sectors per transfer.

The Auto option remains on the menu to allow database-style applications to get the maximum performance from the drive.

10. FPIO and FPIO4

FPIO (Fast Programmed Input/Output) is an enhancement of **PIO**, a means of data transfer that requires the use of the CPU.

This feature supports Mode 3 and a drive-cycle time of 180 nanoseconds. Installing FPIO4 supports Mode 4 and a cycle time of 120 nanoseconds. These modes are compatible with the previous AT Modes 0, 1, and 2. Drives report which mode they support during Autotyping.

FPIO and FPIO4 require 3 bits of CMOS and a small amount of code for both POST and run-time.

11. FDMA

FDMA, Fast Direct Memory Access, is an enhancement of **DMA**, Direct Memory Access, a means of data transfer that does not require the use of the CPU. Also called **Multiword DMA Mode 1**, it supports hard-disk transfer rates of 13 Mb per second, and is compatible with the previous Multiword DMA Mode 0.

12. POST Auto

Reading the drive ID data, POST Auto sets:

- Drive geometry
- Transfer mode (if FPIO or FDMA is enabled)
- Block size (if block or PIO is enabled)
- LBA (if INT13Extensions is enabled).

13. CD ROM Boot

Supports the *Bootable CD ROM Format Specification Version 1.0*, published by Phoenix Technologies and IBM Corp, 14 November 1994. This feature boots and reads from a CD ROM formatted either as a floppy or hard disk as specified.

If **MultiBoot III** is installed, this feature displays the CD ROM as bootable device on the Boot First screen.

14. Predefined Tables

With Release 6 of PhoenixBIOS 4.0, the **pre-defined hard-disk tables** in Setup are an installable feature. Not installing pre-defined tables saves approximately 200 bytes of ROM space.

15. SMART

The **Self-Monitoring Analysis-Reporting Technology** feature, which informs the system that a catastrophic IDE failure is imminent. During POST, the BIOS enables SMART and checks for error conditions using new technology now available in hard-disk drives. Each time a drive is Power Managed, the BIOS checks SMART status. If it finds a failure prediction, it saves the error and displays it at next boot-up.

If it detects an imminent failure, the BIOS issues the message:

IDE Failure Prediction.

A new field in Setup enables the end user to disable this feature.

16. ARM

ATAPI Removable Media (ARM) supports drives for high-capacity floppies that can be formatted as floppies or hard disks, e.g., MKE LS 120.

17. IRM

IDE Removable Media (IRM) supports drives for high-capacity floppies that can be formatted as floppies or hard disks, e.g., Iomega Zip and Fugitsu MO.

18. ULTRADMA

UltraDMA supports 33 MB/sec hard-disk transfers, ATA/33 (UDMA Mode 2), ATA/66 (UDMA Modes 3 and 4), and ATA/100 (UDMA mode 5).

19. PREDELAY

Supports end-user-selectable delay period before the detection of a drive during POST. Accommodates detection of slower drives. Requires 6.0 Setup.

20. OTHERATAPI

Supports ATAPI devices not supported by other HDD features. Support includes PIO, FDMA, and ULTRADMA.

21. I²O

Supports I²O for hard disks. See "Phoenix Intelligent I/O" above.

22. DMACAPABLE

Provides FPIO, FDMA, and UDMA support for ATA and ATAPI devices required by PIIX4 App Note 5.

23. HIDE PTABLE

Supports enabling and disabling the partition-hiding feature of the Iomega Zip drive.

24. RM SOFT RESET

Issues a soft reset to removable-media drives when INT 13h is invoked with AH = 0 and DL = 0. Without this option, all IDE drives receive a hard reset.

25. USB Mass-Storage

Supports booting from a USB floppy, hard disk, or CD ROM.

26. CBI Transport

Supports USB devices that communicate with the Control-Bulk-Interrupt protocol.

z. 1394 Mass Storage

Supports booting from a 1394 (FireWire) storage media.

C. PC Cards (PCMCIA)

Phoenix Technologies develops PC-Card configuration and management software for a number of operating systems and platforms. PhoenixBIOS supports booting off a PCMCIA card (using MultiBoot III). Phoenix also offers the following PC Card products:

- **Bay Swap, Bay Swap NT, and Bay Swap Plus** for warm and hot swapping of devices in system bays. In portables, such devices are battery, floppy, and ATAPI-class devices. Desktop systems include ATAPI-class devices.
- **Card Executive for NT** for hot insertion and removal of I/O cards under Windows NT 4.0, including specified CardBus cards, ATA cards, CD-ROMs, Modems, LAN cards, and Multifunction cards.

1. Overview

PC Cards are credit-card sized peripheral devices of standardized height, length, width, and electrical dimensions. All PC Cards are of the same width (54mm) and they all have a 68-pin connector on the back of the card. The height or thickness dimensions of PC Cards vary. The variation in thickness of PC Cards is referred to as its Type. There are three Types of PC Cards classified by roman numerals - Type I, Type II and Type III. Type I cards are the thinnest and are exactly 3.3mm thick. Type II cards are 5.0mm thick, and Type III cards are 10.5mm thick. All three of the card types use the same connector allowing thinner cards to be inserted into a thicker slot. All three types of cards are normally the same length; however, there are extended cards on the market that are longer and protrude out of the slot on the PC enabling the addition of wireless capabilities or other large I/O connectors.

PC Cards are inserted into designated slots in the exterior chassis of personal computers. Most computers can support the use of more than one PC Card simultaneously and often have a stacked slot configuration, which allows two PC Cards to be inserted one on top of the other separated by narrow carrier housing. Once a PC Card is inserted into one of its target slots, an electrical connection is made between the 68-pin host connector on the PC and the 68-pin connector on the PC Card.

PC Cards can be inserted while the PC is on or off. When a PC Card is inserted into a slot in a PC, a series of electrical messages are sent to a chip, called a PC Card host controller, which resided on the PC. The electrical messages are interpreted by a series of software on the host PC called Socket Services and Card Services. The messages are conveyed to the operating system (such as Windows 95) and to any other software application, which uses information provided by the PC Card. It is this intricate communication between hardware and software, which makes PC Card technology a challenging, and interesting interconnect to support.

The [PCMCIA](#) (Personal Computer Memory Card International Association) is the industry organization that develops and publishes standard specifications for PC Card. The PCMCIA was formed in early 1990's and have published three major revisions of the PC Card Standard - the most current of which was published in May, 1997 and updated in June, 1998.

Phoenix Technologies has been a very active contributor and participant in both the PCMCIA and in the creation of PC Card Standards. Phoenix is an executive level member in the organization, at the highest level possible, allowing the company to foster growth and compatibility in the PC Card industry. In the past 7 years, Phoenix has held a position on the PCMCIA Board of Directors for three years. Phoenix has chaired and co-chaired several PCMCIA Working Groups, including Card and Socket Services, and marketing committees, and has financially supported the activities of the PCMCIA.

2. Device Types

PC Cards are unique peripherals because a multitude of different peripheral functions can be housed in the format of a PC Card. The best way to think about this is that the PC Card is just a shell - what is inside the shell is the particular function needed for a particular job. While the card Type does not always indicate the function that the card is capable of, there are functions that are best suited to a particular card Type. For example, most memory cards (cards which can store data like a diskette or zip drive) are Type I or Type II cards. Most modem cards (cards that allow you to connect directly to a phone line and send faxes or log onto the internet) and network cards (cards which allow you to connect to a company network server) are Type II. Type III cards are primarily used for mass storage, like a miniature hard disk drive. Just like their larger peripheral cousins, each function on a PC Card has its own unique attributes. In other words, a PC Card modem can be of various speeds (28.8 bps or 56K bps) just like a large, external computer modem.

Recently, industry introduced **Zoomed Video (ZV)**, and **CardBus**. Zoomed Video PC Cards are multimedia capable cards, typically allowing large streams of video and audio data to be piped through the card. ZV cards allow users to watch full-length MPEG or MPEG II movies right from their PC. The second relatively new card type is called CardBus. CardBus cards significantly speed up the transfer of data from an external source such as a network to the PC.

Another new addition to the PC Card family is a series of smaller cards, also called half-size cards, which can be plugged into smaller devices like digital cameras, hand-held information appliances, or even mini digital cassette recorders. These cards are also standardized by the PCMCIA.

PC Card Technology is the most mature of all Phoenix supported interconnects in terms of industry adoption and availability. PC Cards have been shipping in large quantities since 1994. Today, 100% of all portable computers sold have one or more PC Card slots.

Unlike some of the newer interconnect technologies, PC Cards are supported by operating-system vendors, PC Card manufacturers, and third-party software suppliers. Microsoft started supporting the use of PC Cards in Windows® 95 by building in PC Card configuration software into the operating system. Current versions of Windows and Windows NT also support cards without the addition of third-party card and socket services software. The availability of software and hardware support for PC Cards ensures that this technology will continue to be widely accepted by users.

VI. PhoenixBIOS Boot Features

A. Phoenix MultiBoot III™

Phoenix **MultiBoot III** is an implementation of the BIOS Boot Specification developed by Phoenix, Intel, and Compaq. It identifies all **IPL (Initial Program Load)** devices in the system and attempts to boot them in the order specified in Setup. MultiBoot III accommodates devices from which an operating system can be booted, such as:

- Floppy Drives, including 120 and 3-Mode floppies
- IDE and SCSI fixed disks
- ATAPI CD-ROMs
- Network cards (remote boot)
- PC Cards (PCMCIA)
- USB storage media
- Iomega Zip Drives
- 1394 (FireWire) storage media

MultiBoot III employs a boot scheme that's generic and flexible enough to boot from virtually any current or future device.

MultiBoot III features a Setup **Boot Menu** with the IPL data on a single Setup page.

There are other new features for enabling or disabling boot devices in Setup, allowing devices to remain in the boot table even when removed from the system, and scaling MultiBoot features up or down to meet space requirements.

MultiBoot III can also boot from PnP ISA and PCI devices that have option ROMs with a valid expansion header.

B. Phoenix QuietBoot

As the computer industry turns toward serving customers with less computer experience, **QuietBoot 600** replaces the customary technical messages during POST with a more visually pleasing and comfortable display (**OEM screen**).

During POST, right after the initialization of VGA, **QuietBoot** displays an illustration called the **OEM screen** during system boot instead of the traditional **POST screen** that displays the normal diagnostic messages.

The OEM screen has these features:

- A default illustration which the OEM can replace by using the bit-mapped-logo conversion utility distributed with QuietBoot 600, **BMP2BIN.COM**
- Optional **Fade-in** and **Fade out**.
- An optional prompt for switching to the POST screen or Setup.
- Provision for optional **screen updating**.
- An optional and customizable **Progress Meter**, that indicates the progress of POST.
- A new option, **VSA Logo**, supports a 256-color OEM screen

The following describes the behaviors of the OEM screen.

To access Setup, display the MultiBoot menu, or simply display the POST diagnostic messages, you can simply press one of the hot keys described below. The defaults assume that most end users don't want to be bothered with diagnostic information.

The OEM screen stays up until just before the operating system loads unless:

- You press <Esc> to display the POST screen.
- You press <F2> to enter Setup.
- POST issues an error message.
- The BIOS or an option ROM requests keyboard input.

C. Phoenix QuickBoot

QuickBoot significantly shortens the time to boot by skipping certain tasks during POST and enabling special speed features. End users can enable and disable QuickBoot in Setup.

The speed of QuickBoot largely depends on the system's configuration and the particular algorithm used for BIOS compression. The following chart shows QuickBoot speed using two different algorithms on a system with a 430HX motherboard with 32MB of DRAM, 133 MHz P54C CPU, logo, and a boot block with the IDE channels disabled (excluding video BIOS initialization time).

Boot failure, bad CMOS, or any other errors detected during POST disables QuickBoot for the next boot, even if QuickBoot is enabled in Setup. Servers do not require QuickBoot.

D. Setup Boot-Menu Features

- Selectable Boot Sequence
- Floppy seek disable.
- Summary Screen.
- Enable/Disable Setup Prompt.

VII. PhoenixBIOS POST Features

Almost immediately after power-on, the BIOS conducts the Power-On Self Tests (POST) during which it tests and initializes the devices required to boot the operating system. POST Services.

A. POST Services

Source-code customers can customize POST tasks by creating **hook routines** that execute either before or instead of the normally scheduled POST tasks.

PhoenixBIOS **POST Services** isolate different POST activities from one another, greatly simplifying the addition of new tasks. POST Services include:

- **Phoenix Dispatch Manager.** Manages POST decompression, loading, of all the other POST Services and communication between them.
- **NVRAM Manager** now uses identical CMOS access for both BIOS and Setup. See below for more information.
- **POST Memory Manager** dynamically allocates memory during POST, solving resource conflicts.
- **POST Display Manager** handles all screen displays during POST. All POST strings now accessed from a database separate from BIOS code.
- **POST Error Manager** isolates error handling from hardware and features selected. Processes all POST errors and displays them at the end of POST.

B. The NVRAM Manager

During late POST, the BIOS reads the values stored in Non-Volatile RAM (NVRAM) and writes them to the chipset registers. These NVRAM values can be one of the following:

- An end-user Setup Selection stored in NVRAM
- The Setup Default, which can be:
 - Loaded into Setup by <F9> and then stored in NVRAM by the end user.
 - Loaded directly into NVRAM by a bad checksum.
 - Loaded directly into NVRAM by Auto Configuration, which selects defaults according to CPU type, speed, and stepping.

NVRAM serves these purposes:

- It allows end-users to use the Setup utility to modify the system parameters, thus changing the subsequent operation of the PC.
- It allows Auto Configuration to change the chipset settings if there has been a change in the CPU.
- It retains the configuration information when the PC is turned off.

The most common NVRAM in use is battery-backed CMOS RAM. Most IBM-compatible systems have some standard CMOS RAM (128 kB) and many systems have additional (extended) CMOS RAM (256 kB). Any non-volatile media can be used as NVRAM. PhoenixBIOS supports up to four banks of NVRAM; each bank is referred to as a **media type**. Each media type has its own access routines, providing platforms the option of supporting a mix of physical NVRAM types. PhoenixBIOS 4.0 supports (i.e., has access routines for) CMOS RAM located on the same chip that contains the Real-Time Clock.

Any system parameter that is editable by Setup or other utility must have storage space allocated in NVRAM. PhoenixBIOS provides the hardware-independent **NVRAM Manager** for allocating, mapping, and accessing NVRAM. This method eliminates the need to manually allocate and track NVRAM fields.

The NVRAM Manager offers these advantages:

- Optional BIOS features and enhancements that require NVRAM storage can be developed without concern for NVRAM conflicts occurring between those features.
- Source-code engineers can easily adapt BIOS code for different forms of NVRAM.
- Automates the allocation and access of NVRAM. The NVRAM Manager creates an NVRAM memory map. Engineers can create a fixed NVRAM memory map if needed, or a partially fixed/partially dynamic map, which the NVRAM Manager will fill in with a best-fit algorithm.

Since it is hardware independent, the NVRAM Manager is capable of managing multiple NVRAM types simultaneously (e.g. CMOS, extended CMOS, EEPROM, fixed disk, etc.).

C. POST Configuration Features

One of the chief functions of the BIOS during the Power-On Self Test (POST) is to conduct an inventory of devices, on the motherboard and to assign them resources system resources as required. System resources include:

1. DRAM memory space
2. Interrupt Request (IRQ) lines
3. Input/Output addresses
4. Direct Memory Access (DMA) channels and Ultra Direct Memory Access (UDMA) channels for PCI devices.

Assigning resources to a device is called **configuration**. The POST configuration features include:

1. Auto Configuration

When installed and enabled in Setup, **Auto Configuration** always loads Setup Defaults into the chipset registers during POST without end-user intervention. Source-code engineers can install Auto Configuration on a register-by-register basis.

2. Auto Initialization

If Auto Configuration is not installed, or if it is disabled in Setup, and if CMOS is good, **Auto Initialization** (the default) loads CMOS values into registers. If CMOS is bad, it loads the Setup Defaults into the registers and displays a notice of the correction.

3. PCI-IRQ Routing

In case of a conflict between devices claiming the same resources during POST, the BIOS must resolve the conflict. For example, the **Conflict Detection and Resolution (CDR)** algorithm in POST uses **IRQ Routing Tables** in the BIOS determine which IRQ to assign to which PCI device.

PCI interrupts are routed through the chipset to the **Peripheral Interrupt Controller (PIC)** and are level-triggered. Most PCI chipsets provide for **PCI-IRQ routing**, a programmable mechanism using chipset registers for routing interrupts from PCI devices (embedded devices or slots) to IRQ lines (0 to 15) on the PIC.

Most PCI devices don't care which IRQ they use, and the PnP software assigns them automatically. Exceptions such as PCI IDE devices, however, require the use of a particular IRQ.

PCI devices can have up to four physical interrupt lines, INTA to INTD. Single function devices always use INTA. On multi-function devices, these four lines can be assigned to different functions of the device. More than one function can be assigned to the same interrupt line.

Furthermore, since the PCI devices can share the same interrupt lines, different functions from different devices can share the same interrupt line. Different interrupt lines can share IRQs on the PIC.

4. Plug-and-Play Configuration

To secure a conflict-free allocation of resources, a BIOS with Plug and Play uses the following resources:

- The Setup defaults stored in the BIOS code.
- The **Device Nodes**, individual structures with the configuration data for each ISA motherboard device, which includes possible configuration options.
- Lists of possible configuration-options, provided by register data in each Plug-and-Play ISA or PCI device.
- NVRAM configuration data, stored as one of the following:
 - Configuration data stored in **System Configuration Data (SCD)** format, for systems with an EISA bus.
 - The previous session's configuration data for all devices, formatted as **Extended System Configuration Data (ESCD)**.
 - Configuration data for Legacy devices only, formatted as Bitmap.

With information from these sources, the PnP BIOS does the following during POST:

1. Allocates resources for static devices, including static motherboard devices, legacy cards, and EISA devices according to configuration data stored in NVRAM.
2. Dynamically configures the Plug-and-Play ISA and PCI devices as specified. If the system uses a PnP operating system, end users can specify in Setup whether the BIOS configures **all** ISA Plug-and-Play devices or only those devices required to boot the operating system (SCSI, Network cards, etc.). Setting "Plug & Play OS" to "Yes" in Setup gives the PnP operating system more control over devices and drivers loaded. A PnP OS configures those devices not configured by the BIOS in POST.
3. Records any new configuration data in NVRAM in ESCD/SCD/Bitmap format if selected.

VIII. PhoenixBIOS Setup Features

During POST, the end user can press <F2> to enter Setup and change the system parameters originally specified in the BIOS defaults. The purpose of Setup is the following:

- Change system hardware (e.g., add a new drive)
- Change system behavior (e.g., enable password protection)
- Optimize system performance (e.g., change disk access)
- Store the user's selections in CMOS (non-volatile memory)

During POST, the BIOS will access the information specified in Setup and stored in CMOS and use it to configure the hardware and to manage the system's performance.

PhoenixBIOS has a **Setup Engine** that is responsible for building the Setup screens when requested. The contents of those screens are now organized and defined in a template, which also defines the contents of the Boot Menu, Password Prompts, Summary Screen, and Backup and Virus-Check reminders, as well as specifying the Setup language, the line-draw characters, and the strings used in the legend and titles. The template references **Setup Nodes**, each of which defines a single item on a Setup screen.

All POST and Setup strings are managed in a single module, which can be processed and updated without having to build the rest of the BIOS. The String Database is a collection of strings referenced by a language code and a label. The string database contains all the display and content data used by the BIOS. Items in it are referenced by Setup Nodes for the display of Setup Menus. It also contains items used for other BIOS displays such as version and OEM information.

For a description of the PhoenixBIOS Setup, see the *PhoenixBIOS 4.0 User's Manual*.

IX. PhoenixBIOS Security Features

A. Setup Security-Menu Features

The following security features governed the internal BIOS regime specified by the end-user in the Setup Security menu. Those menu items gave the end user password control of access to:

- Two-Level Password Control.
- PS/2 Style Password Control.
- Quick Lock FDD Access Control.
- HDD Boot Sector Protection.
- Backup Reminder Message.
- Virus Check Reminder Message.

See the *PhoenixBIOS 4.0 User's Manual* for more information.

B. BIOS Security Services

Because **external clients** must access security information or functionality only the BIOS can provide, PhoenixBIOS provides **BIOS Security Services** for both internal clients (such as Setup nodes) and external clients not linked with the BIOS.

This new technology uses two tables, one that defines security states and the other that defines the permissions under which access to a device is allowed. It is also possible to govern individual Setup items with separate security provisions.

The BIOS Security Services provide a mechanism for external clients to extract information from the BIOS or instruct the BIOS to perform a specific function.

X. PhoenixBIOS Power-Management Support

The extensive power-management software available in all Phoenix BIOS products is based on its long experience and leadership in developing energy-saving technologies for portable PCs.

PhoenixBIOS power-management includes these products:

- **Phoenix Miser 4.0**, BIOS-based power management.
- BIOS support for **Advanced Power Management (APM)**
- BIOS support for **Advanced Configuration and Power Interface (ACPI)**
- **Phoenix Power Suite™** utilities for managing battery use in portable PCs.
- **Phoenix Save-To-Disk**, for saving the current state of the computer to disk.

A. Phoenix Miser 4.0

Miser 4.0 is a platform-configurable, BIOS-based, Run-Time power management that provides full Energy Star support for power management of CPU, hard disk, monitor, and any combination of power-managed devices. Supports APM 1.2 (Advanced Power Management) interface.

Miser 4.0 provides the following services:

1. Initializes power-management devices and routines during POST.
2. Utilizes chipset inactivity timers to monitor specified power-management events, such as keyboard inactivity timeouts.
3. Changes the power-management states of specific devices in response to specified power-management events, for example, by turning off the disk drive in response to a completed period of inactivity.
4. Provides a window in Setup in which the end user can specify power-management options.

B. Phoenix APM

Advanced Power Management (APM), supported in PhoenixBIOS Miser 4.0, is a method of power management that requires the cooperation between the BIOS and the operating system. It is based on a Microsoft 1993 specification for power managing in earlier versions of Windows. APM has its own interface for DOS and Windows, with which end users can enable or disable power management, overriding the setting specified in Setup.

APM increases the effectiveness of System Power Management by:

- Monitoring CPU activity.
- Managing communications between the BIOS and applications and device drivers that are APM aware.

Phoenix APM supports these states:

- Full Power
- Idle
- Standby
- Suspend

Phoenix APM provides the interface between the power-managed components of the motherboard and the **APM driver**. The APM driver monitors CPU activity and manages communications between the BIOS and APM-aware device drivers and applications. The APM-aware device drivers manage the power of individual add-on devices.

As soon as the APM driver loads during bootup, it uses an APM BIOS function call to periodically query the BIOS concerning **power management events**. If an event has taken place, the function call returns an **event code**, which informs the APM driver what to do.

In the event of a standby or suspend request, the APM driver gives APM-aware applications and device drivers the opportunity to prepare for or reject the request. The APM driver puts a hold on any action by the BIOS until it receives the required clearance from all APM-aware applications and device drivers. Only then does it call the **Set Power State** function to either authorize or reject the request.

The APM driver can also send the BIOS information about CPU activity and issue **CPU Idle** and **CPU Busy** calls. The BIOS may respond by stopping the CPU with a HLT command. Any activity interrupt wakes up a Halted CPU, making it unnecessary to process the CPU Busy call.

As with System Power Management, certain interrupts such as a modem ring or a real-time clock alarm can restore the system to Full Power from other states.

C. Phoenix ACPI BIOS

The **Advanced Configuration and Power Interface (ACPI)** specification gives an ACPI operating system—such as Windows 98 or 2000—power management as well as Plug-and-Play configuration for notebooks, desktops, and servers. ACPI is the keystone in Microsoft's Operating System Directed Power Management (OSPM). OSPM gives operating system more authority about when to do power management and relies on the BIOS to determine how to do it.

During POST, the BIOS creates special ACPI tables in memory detailing the power-management and configuration capabilities of the hardware devices. The operating system later uses these tables in a wide range of configuration and power-management services.

The Phoenix ACPI BIOS provides not only a robust core implementation of the ACPI specifications but also these important timesaving engineering features:

- The **Phoenix ACPI Architect™**, a Windows-based utility for solving the most common ACPI porting issues.
- **Phoenix MCD2ACPI.EXE**, a utility that optimizes the information provided by **Motherboard Configurable Devices (MCD)**.
- **Phoenix MCD Services**, which provides ACPI management of many device not strictly compliant with the specification.

The main advantages of ACPI are an open architecture and a more cooperative Plug and Play environment. The introduction of ACPI for the first time brings the full benefits of power management to Windows NT. Windows NT 5.0, Certification for Windows and Server 2000, PC 2001, and OnNow require ACPI.

For more information about ACPI, see ACPI in the Technologies section.

D. Phoenix Save-To-Disk

Save to Disk is a program that saves the state of the computer to the hard disk drive when it enters SUSPEND mode. The state includes the contents of video and system and keyboard RAM, SMRAM, and all system registers. After a Save to Disk occurs, the system is automatically powered off. When the system is turned on, the operating system and all applications that were active before the SUSPEND are restored, and the computer is ready to use in a fraction of the time required to reboot the system.

PhoenixBIOS Save-to-Disk includes:

- **User-selectable suspend to RAM** - You can select what to do on timeout from the Setup Screen, Suspend or Save to Disk. An OEM can also program the system to activate Save to Disk when a low battery condition is detected.
- **Partition and DOS File Save to Disk Support** - Save to Disk data can be written to either a dedicated partition or a hidden DOS file.
- **High Speed Non-SMI Mode Processing** - After Save to Disk code is swapped into the cacheable Base 640KByte memory, the CPU exits SMM (System Management Mode) and executes Save to Disk in REAL mode.
- **Quick Chipset Porting** - Save to Disk ports faster to new platforms for two main reasons: the number of chipset-specific requirements has been reduced because Save to Disk code executes outside of SMM; and, the chipset-specific code and data are located in a single file that is separate from the core.
- **Quick Save Support** - Save to Disk does not save blocks of extended memory that are zeroed out. It sets a bit in a D/Z table to indicate these blocks are zeroed out. When the system is resumed, it stores zeros in this memory instead of reading the information from the hard disk.
- **Quick Memory Configuration** - The time required by the Resume from Disk operation has been reduced by restoring the system's DRAM and cache configuration from the CMOS RAM, instead of calling the BIOS RAM and cache auto sizing routine(s).

PHDISK.EXE is the hard-disk preparation utility for Save-to-Disk. It can prepare either a dedicated partition or an MS-DOS-based hidden file prior to storing system configuration data, and system and video memory.

XI. PhoenixBIOS System-Management Features

A. Phoenix SMBIOS 2.3.1

System Management BIOS (SMBIOS) is the latest revision of the **Desktop Management Interface BIOS (DMI BIOS)** specification. This feature provides a standard method for system managers and software to inventory current and possible PC configurations either locally or remotely, through a network.

The contribution of the BIOS to a DMI system is to build in memory the **SMBIOS Data structures**, which contain information about the BIOS and the motherboard along with components installed on the motherboard. For details, see the *System Management BIOS Specification*.

B. Phoenix IPMI

The **Intelligent Platform Management Interface (IPMI)** defines the interfaces for extending platform management between boards within the main chassis and between multiple chassis. The Intelligent Platform Management Bus (IPMB) is an I²C-based bus that provides a standardized interconnection between different boards within the chassis. Multiple chassis are controlled through the **Intelligent Chassis Management Bus (ICMB)**.

IPMI uses the IPMB to take inventory of any system sensors, monitor their status, and log errors. Applications can query the status of the system using the IPMI and determine if any of the sensors are recording out-of-bounds conditions in the log.

At the center of the IPMI is a micro-controller called the Baseboard Management Controller (BMC). The BMC performs the following functions:

- Manages the interface between system management software and the platform management hardware

- Provides autonomous monitoring, event logging, and recovery
- Serves as a gateway between system management software and the IPMB and ICMB.

The BMC communicates to the IPMB and the ICMB with message-based interfaces using a request and response protocol. A request message, also called a command, is matched by a response message, but multiple request messages can be interleaved with multiple response messages on the bus. The BMC and the bus devices manage their queue of pending responses and commands.

C. Phoenix Universal Console Redirect (UCR)

Expanded **Universal Console Redirection (UCR)** is an optional installable feature for monitoring POST messages and running Setup and an operating system from a remote serial terminal.

The PhoenixBIOS UCR features:

- Simple installation.
- Setup selection of I/O Port, using a motherboard I/O device for local serial connection.
- Setup selection of baud.
- Setup selection of handshaking by hardware, software (XON/XOFF), or none.
- Setup selection of direct or modem connection.
- System video output replicated on terminal.
- Remote terminal keyboard input deposited in system keyboard buffer.

The operators at the other machine have the same controls (e.g., using the "hot" keys and rebooting the system during POST). They also have the ability to enter SETUP and modify the settings stored in CMOS. The system-security features however, cannot be circumvented if enabled.

The current limitation is that most operating systems other than MS-DOS interfere with the console redirection, disabling access by the other machine after POST. The reason for this is that the UCR reserves the serial port for redirection use. The port chosen by the user in SETUP is removed from the BDA (BIOS Data Area). Any operating system that does **not** rely on the BDA to get information on the available serial ports, but instead, independently searches for them, will cause the connection to be lost after POST. This event is true for all applications, including mouse drivers.

D. Phoenix Wired for Management (WfM)

Wired for Management is an Intel's initiative to make PCs universally manageable and universally managed. WfM provides guidelines for a new generation of platforms that can be centrally managed over networks to reduce Total Cost of Ownership (TCO). This ability of managing systems over the wire benefits in the following areas:

- **Asset Management**—Built-in instrumentation identifies aids in managing and keeping track of static and dynamic data related to both software and hardware components in the system. Uses industry standard interfaces like DMI, CIM.
- **Universal Network Boot**—The Preboot Execution Environment (PXE) technology makes it possible to control a system remotely, even on OS-absent platforms. The PXE agent on the system allows it to interact with a remote server to dynamically retrieve the requested boot image across the network, making it possible to
 - Remotely install an operating system
 - Remotely configure a new system (Remote Setup)
 - Remotely run diagnostics
 - Remotely and securely update flash (BIOS)

The remote operations enabled by the Preboot Execution Environment can lower the costs of administration and technical support, thus decreasing TCO.

- **Off-Hours Maintenance and Power Savings**—The system can be automatically awakened or transitioned from a sleep state to a fully powered state, over the network. Once awakened, the system can be directed to run utilities such as virus scan or disk backup, or install software upgrades, and then return to a sleep state.

The IS manager can use the remote wake-up feature, combined with remote boot and other remote control capabilities, to run management tasks remotely during off-hours.

WfM 2.0 Baseline specification defines the minimum level of power-management capabilities of the systems and to build manageable systems.

- **System Diagnosis and Repair**—Management applications could use standardized trouble tickets to receive problem descriptions by a managed system and their users. Using the Problem Resolution Software, the management application can also provide a resolution to the problem and verify that the particular solution is satisfactory.

Management applications can also use PXE to help resolve problems. For example, an IS Manager can take remote control of the system to diagnose and repair it.

- **Investment Protection**—With the help of built-in instrumentation and management applications capable of cross-mapping CIM and DMI data, IS department can keep track of various systems' configuration thus aiding investment protection.

To support these goals, **Phoenix Wired for Management** supports these three technologies, which can be installed as separate features:

- **Pre-Boot Extension Environment (PXE)**, for remotely accessing a system's resources even in the absence of a functioning Operating System.
- **Remote Lockout**, for preventing interruption of a critical remote operation because of an end user's moving the mouse or pressing a key or a reset button.
- **Wake On LAN**, for waking up a remote system from a sleeping or power-off state in order to perform a remote operation.

XII. PhoenixBIOS Build Features and Utilities

A. PhoenixBIOS ROM Maker Tools

Included in the core of Phoenix BIOS products are modular technologies that greatly simplify the engineering tasks of porting of the BIOS software to a new platform.

ROM Maker Tools use scripts to build the BIOS source code and turn it into a **module** that can be compressed and put into a ROM image along with other ROM modules. A **ROM module** can be any file that contains code (such as the system BIOS, an option ROM, or VGA BIOS), OEM text, or a graphic image). The final **ROM image** is the concatenated binary file that includes all the ROM modules necessary for the target system to function correctly.

This method:

- Separates the ROM image into independent modules that can be relocated in system (as opposed to shadow) memory.
- Replaces one or more modules in a flash part without reprogramming the entire ROM image, e.g. replace a VGA BIOS or the Setup strings.
- Compresses code for Flash parts greater than 128 kB.
- Accommodates Flash parts with 16 kB boot blocks.
- Inserts fiduciary marks into specified locations of a ROM image.

B. PhoenixBIOS Deployment Utilities

1. Phoenix 32-bit Debugger

The Phoenix 32-Bit Debugger is a Windows-based utility for analyzing and debugging pre-boot, Run-Time, and SMI code designed for PhoenixBIOS 4.0. It uses either serial or bi-directional parallel cables connecting the host computer and the development platform.

The debugging Setup includes these three items:

- A host computer running the Phoenix 32-Bit Debugger for Windows.
- A development platform running PhoenixBIOS 4.0 with the debug feature installed in the BIOS.
- A communication system between the two machines.

When you restart the BIOS on the development platform, the Debugger running on the host displays a listing of the Test-Point codes as they are executed, stopping at previously defined **breaks**.

With the BIOS execution stopped, you can:

- Set up new breaks and continue or restart the BIOS from the Debugger.
- Step through and examine the unassembled code and the related routine labels.
- Search for routine labels, set them for breaks, and display related object, source, and unassembled code.
- Examine and modify related registers, stack, flags, I/O ports, etc.
- Examine and modify memory, segments, and registers used by PnP, PDM, POST, Setup, CMOS, the CPU, etc.
- Continue BIOS execution to the next break code or to the end of POST.

By running BREAKIT.EXE on the development platform, you can continue debugging BIOS code during Run Time, including MISER and SMI code.

The Debugger makes use of several kinds of breaks:

- **Break Codes.** There are two kinds of break codes: those specified in TPOINTS.ASM and others in **Break-Code macros.** Both of these use a one-byte hex number specified in the source before the Build. In the Debugger, you can set only one break code at a time.
- **Break Addresses** (using DR0 and INT 1), which are set in the Debugger and take place when a memory or an I/O address is accessed. In the Debugger, you can set only one break address at a time. You can specify a symbolic (source code) label to set a break address.
- **Break Points** (using INT 3), which you can toggle in the Debugger by selecting an instruction in the Code window and pressing <Ctrl-F9> (as long as the instruction is writeable in RAM or write-enabled Shadow). You can set up to 128 break points at a time.
- **Break Sequences**, which are set in the Debugger and define multiple conditions for a break using both break addresses and break codes.

Break Codes use one of the Debug registers as a scratch pad to pass control to the Debugger by a jump that is part of the BREAK_POINT macro.

Break Addresses and single stepping use the **Set Break Address** instruction and single stepping of 386+ CPUs. When single-stepping or reaching a Set Break Address, the CPU issues **INT 1**, hooking the Debugger on the host, which takes control of the development platform at that point. The Break Address is a 32-bit linear address within 4 GB memory. You can set the BIOS to break on the execution, read, or write of a memory address, or on the access of an I/O address. Because address breaks use an INT 1 vector, memory must be ready.

Break Points use another mechanism. When you set a break point in the Code Window of the Debugger with <Ctrl-F9>, the Debugger replaces the original instruction in the BIOS with an INT 3 instruction and saves the original instruction. When the BIOS reaches that point, the INT 3 stops execution. The Debugger then restores the original instruction, which is the first to be executed after the break.

The Debugger features a number of windows, including:

- The **Code window**, for single stepping and tracing through code and symbols, displaying and modifying CPU registers and flags and the stack, and for setting break points and break addresses.
- Up to four **Data windows**, for displaying and modifying memory.
- The **Source** window, for displaying the source code related to the label symbols on the Code window using ATAGS.
- The **Break Condition** window, for setting code and address breaks along with register conditions.
- The **Sequence Break** window, for setting a number of ANDed conditions for breaking the BIOS.
- The **Access** window, for displaying and changing register data.
- The **I/O** window, for displaying and changing the status of I/O ports.
- The **NVRAM** window, for displaying and changing NVRAM tokens.
- The **CPU Register** window, for displaying and changing CPU registers.
- The **BCP Analyst** window, for displaying BCP information.
- The **PnP Analyst** window, for displaying the PnP table structure during POST.
- The **PDM Analyst** window, for displaying the modules serviced by the Phoenix Dispatch Manager.
- The **POST Analyst** window, for displaying information from the POST cold or warm-boot tables.
- The **Register Table Analyst**, for displaying all the register tables needed to verify a chipset port.

2. Phoenix Access

Phoenix Access is a general-purpose editor for accessing, displaying, editing, and writing to hardware addresses such as RAM, chipset registers, and CMOS. The executable, ACCESS.EXE, uses a text file from which it parses and stores into memory information about ports, indices to set, unlocking procedures, and other necessary details for editing.

Access has number of features beyond simply read and writing to hardware. A screen of registers, where a register is RAM location, CMOS location, or chip set register, can be saved to file of the users' choice. The current register values in a screen can be compared with values saved in a file. Register values will update periodically without requiring the user request the utility to read the register. DOS Access can be run as a TSR and popped up under the Dos shell, Debug, or other DOS programs. CMOS checksumming, bit information, and linking text to register are also available.

3. Phoenix SymCMOS

Manufacturing lines and remote management over a network need something besides Setup to configure a personal computer.

The application utility SymCMOS addresses this problem by providing a Run-Time method for reporting and configuring the non-volatile CMOS settings of a personal computer.

The two primary functions of SymCMOS are:

- Read from CMOS the current configuration values and write them to a report file.
- Read from an input file and write to CMOS a specific set of configuration values.

Since the report and input files use the same format, you can use the report file from one system to write values to another system, or you can edit the report file and write it back to the same system.

You can use the utility to:

- Read the symbolic names used to identify Setup fields and write them to a report file. The field names include the names of the parent items.
- Determine the type of a Setup field and the legal options for that field, which could be a range of symbolic names or numeric values.
- Determine the current value of a Setup field.
- Determine the default value of a Setup field.
- Set new values in CMOS and in Setup fields.
- Determine and use the language currently selected in Setup by reading the Language Index Value in CMOS.
- Combine the contents of NODES.ROM and STRINGS.ROM into a single file, COMBINE.ROM, for future use by the utility.

4. Other Deployment Utilities

Packaged with the PhoenixBIOS source code are these other programming utilities:

- **PRT.EXE** displays the PCI IRQ Routing Table
- **PNPBTST.EXE** executes and tests the BIOS **PnP Run-Time Services**.
- **DMIBTST.EXE** tests and displays the DMI and SMBIOS structures. This utility does not use the component interface, the service layer, or the MIF file. This utility does a compliance testing on the BIOS to verify whether the required fields of the structures contain valid values.
- **NVRAM.EXE**, the Non-Volatile RAM System Configuration Data Utility is utility for displaying and editing ESCD data in CMOS.

Index

- 1394, 14
- 1394 Mass Storage, 18
- 32-Bit Debugger, 28
- Access, 30
- ACPI, 25
- ACPI Architect, 25
- Advanced Configuration and Power Interface, 25
- APM, 25
- APM driver, 25
- ARM, 17
- ATAPI Removable Media, 17
- Auto Configuration, 22
- Auto Initialization, 22
- Auto-Config, 21
- Backup Reminder, 24
- Bay Swap, 18
- BDA, 5
- BIOS Enhanced Disk Drive Specification, 16
- BIOS Security Services, 24
- BIOS Update, 6
- Block PIO, 17
- BMP2BIN.COM, 20
- Boot Sector Protection, 24
- Boot Sequence, 21
- BootBlock, 7
- Break-Code, 29
- build, 28
- Cache, 10
- Card Executive, 18
- CardBus., 19
- CD ROM boot, 17
- CDR, 22
- CHS, 16
- CMOS, 11, 22
 - value, 21
- compression
 - BootBlock, 7
- Compression
 - QuickBoot, 21
- Conflict Detection and Resolution, 22
- CPU
 - Busy, Miser, 25
 - Idle, Miser, 25
- Cylinder/Head/Sector. See CHS
- Cylinder-Head-Sector, 16
- DDM, 10
- Debugger, 28
- Device Driver Module, 10
- DMA, 8
- DMIBTST.EXE, 30
- DRAM., 10
- Dynamic System Configuration, 8
- EBDA, 5
- EISA bus, 12
- el ports, 13
- Enhanced Parallel Port, 13
- ESCD, 23, 30
 - utility, 30
- event code, Miser, 25
 - event, PM, 24
 - Fast Direct Memory Access. See FDMA
 - FDMA, 17
 - Flash ROM, 6, 11
 - Floppy seek disable, 21
 - FPIO, 17
 - FPIO4, 17
 - HCRM, 15
 - High Capacity Removable Media, 15
 - Host Controller Services, 14
 - I/O Address space, 8
 - I/O component, 8
 - I/O Platform, 9
 - I²O, 18
 - I2O Run-Time Operating System, 9
 - IDE Removable Media, 18
 - Initial Program Load. See IPL
 - INT 1, 29
 - INT 13 Extension APIs, 16
 - INT 13 Extensions, 16
 - INT16h Extensions
 - NuKBC, 6
 - Intel i960 I²O, 9
 - Intelligent Platform Management Interface, 26
 - Interrupt Vector Table, 5
 - ioInitJ, 10
 - IOP, 9
 - IPL, 20
 - IPMI, 26
 - IRM, 18
 - IRQ routing, 22
 - IRQs, 8
 - RTOS, 9, 10
 - ISA, 23
 - ISA, 8
 - ISA bus, 12
 - Keyboard Controller, 6
 - Keyclick, 6
 - LAN card, 9
 - LBA, 16
 - Legacy card, 23
 - Legacy-Free, 15
 - Logical Block Access. See LBA
 - LZSS, 7
 - MCD, 8
 - media type,NVRAM, 22
 - memory, 10
 - Miser
 - event code, 25
 - PM, 24
 - Miser 4.0, 24
 - Motherboard Configurable Devices. See MCD
 - MultiBoot II, 17, 20
 - Multiple Configuration, 6
 - Multiword DMA. See FDMA
 - NuKBC, 6
 - Numlock Power-On State, 6
 - NUMLOCK., 6
 - NVRAM

- manager, 21, Error! Not a valid bookmark in entry on page 22
- media type, 22
- memory map, 22
 - Setup, 21
- NVRAM Manager, 21
- NVRAM.EXE, 30
- OEM
 - screen, 20
- OEM screen, 20
- OHCI, 14, 15
- Open Host Controller Interface, 14, 15
- Operating System Directed Power Management, 25
- Option ROM, 11
- OSPM, 25
- P6Update, 6
- Password Control, 24
- PC-Card, 18
- PCI, 8, 12
 - IRQ routing, 22
- PCI IRQ Routing Table utility, 30
- pciinitJ, 10
- PCMCIA, 18
- PDM
 - Boot Block, 7
- PELE, 15
- Pentium Pro, 6
- Peripheral Component Interconnect, 12
- Peripheral Interrupt Controller, 22
- PHDISK.EXE, 26
- Phoenix Dispatch Manager, 21
- PIC, 22
- PIO, 17
- Plug and Play, 8
- PM
 - event, 25
- PnP, 8
- PnP Analyst, 30
- PnP Run-Time Services, 30
- PNPBTST.EXE, 30
- POST, 22
 - QuietBoot, 20
 - screen, 20
- POST Analyst, 30
- POST Display Manager, 21
- POST Error Manager, 21
- POST Memory Manager, 21
- POST tasks and beep codes, 5
- POSTAuto, 17
- POST-Time Services, 21
- power-management, 24
- pre-defined tables, 17
- progress meter
 - QuietBoot, 20
- PRT.EXE, 30
- Quick Lock Fdd Access Control, 24
- QuickBoot, 21
- QuietBoot, 20
- QuietBoot 600, 20
- Register Table Analyst, 30
- ROM
 - image, 28
 - module, 28
 - ROM Maker Tools, 28
 - routing, PCI IRQ, 22
 - RTC, 11
 - Runtime Services, 8
 - Run-Time Services, 5
 - S2D. See save to disk
 - save to disk, 26
 - SCSI, 9
 - security
 - services, 24
 - Self-Monitoring Analysis-Reporting Technology, 17
 - serial ports, 13
 - Set Break Address, 29
 - Set Power State, 25
 - Setup, 5, 23
 - NVRAM, 21
 - Prompt, 21
 - QuietBoot, 20
 - Setup Device Node, 23
 - shadow memory, 11
 - silicon support, 5
 - SMART, 17
 - SMBIOS, 26
 - Data Structures, 26
 - SMBus, 13
 - SRAM, 10
 - Summary Screen., 21
 - Super I/O, 8
 - SUSPEND mode
 - save to disk, 26
 - suspend to RAM
 - save to disk, 26
 - SymCMOS, 30
 - System Management BIOS, 26
 - System Management Bus, 13
 - Target ID, 10
 - The Legacy Interface, 14
 - TID, 10
 - Typematic Rate and Delay, 6
 - UCR (Universal Console Redirection), 27
 - UHCI, 14
 - UltraDMA, 18
 - Universal Console Redirection (UCR), 27
 - Universal Host Controller Interface, 14
 - Universal Serial Bus, 14
 - Update Loader, 6
 - USB, 14
 - USB mass storage, 18
 - USB Services, 14
 - VESA Universal Memory Architecture, 13
 - Virus Check Reminder, 24
 - VUMA, 13
 - WfM, 27
 - Windows 95, 8
 - Wired for Management, 27
 - Zoomed Video, 19