BIOS

From Wikipedia, the free encyclopedia

In IBM PC compatible computers, the **Basic Input/Output System** (**BIOS**), also known as **System BIOS**, **ROM BIOS** or **PC BIOS** (/ˈbaɪ.oʊs/), is a *de facto* standard defining a firmware interface.[1] The name originated from the Basic Input/Output System used in the CP/M operating system in 1975.[2][3] The BIOS software is built into the PC, and is the first software run by a PC when powered on.

The fundamental purposes of the BIOS are to initialize and test the system hardware components, and to load a bootloader or an operating system from a mass memory device. The BIOS additionally provides abstraction layer for the hardware, i.e. a consistent way for application programs and operating systems to interact with the keyboard, display, and other input/output devices. Variations in the system hardware are hidden by the BIOS from programs that use BIOS services instead of directly accessing the hardware. Modern operating systems ignore the abstraction layer provided by the BIOS and access the hardware components directly.

The BIOS of the original IBM PC/XT had no interactive user interface. Error messages were displayed on the screen, or coded series of sounds were generated to signal errors. Options on the PC and XT were set by switches and jumpers on the main board and on peripheral cards. Modern Wintel-compatible computers provide a setup routine, accessed at system power-up by a particular key sequence. The user can configure hardware options using the keyboard and video display.

BIOS software is stored on a non-volatile ROM chip on the motherboard. It is specifically designed to work with each particular model of computer, interfacing with various devices that make up the complementary chipset of the system. In modern computer systems, the BIOS contents are stored on a flash memory chip so that the contents can be rewritten without removing the chip from the motherboard. This allows BIOS software to be easily upgraded to add new features or fix bugs, but can make the computer vulnerable to BIOS rootkits.

MS-DOS (PC DOS), which was the dominant PC operating system from the early 1980s until the mid 1990s, relied on BIOS services for disk, keyboard, and text display functions. MS Windows NT, Linux, and other protected mode operating systems in general do not use it after loading.

BIOS technology is in transitional process toward the Unified Extensible Firmware Interface (UEFI) since 2010.[4]

Contents

Terminology

The term BIOS (Basic Input/Output System) was invented by Gary Kildall[5] and first appeared in the CP/M operating system in 1975,[2][3][6][7] describing the machine-specific part of CP/M loaded during boot time that interfaces directly with the hardware.[3] (A CP/M machine usually has only a simple boot loader in its ROM.)

Versions of MS-DOS or PC DOS contain a file called variously "IO.SYS", "IBMBIO.COM", "IBMBIO.SYS", or "DRBIOS.SYS"; this file is known as the "DOS BIOS" (aka "DOS I/O System") and contains the lower-level hardware-specific part of the operating system. Together with the underlying hardware-specific, but operating system-independent "System BIOS", which resides in ROM, it represents the analogous to the "CP/M BIOS".

In other types of computers, the terms *boot monitor*, *boot loader*, and *boot ROM* are used instead. Some Sun and PowerPC-based computers use Open Firmware for this purpose.

With the introduction of PS/2 machines, IBM divided the System BIOS into real-mode and protected mode portions. The real-mode portion was meant to provide backward-compatibility with existing operating systems such as DOS, and therefore was named "CBIOS" (for Compatibility BIOS), whereas the "ABIOS" (for Advanced BIOS) provided new interfaces specifically suited for multitasking operating systems such as OS/2.

There are a few alternatives to the functionality of the "Legacy BIOS" in the x86 world:

Extensible Firmware Interface, Open Firmware (used on the OLPC XO-1), and coreboot.

The BIOS boot process

When the x86 processor is reset, it loads its program counter with a fixed address near the top of the 1 megabyte real-mode address space. The address of the BIOS's memory is located such that it will be executed when the computer is first started up. A jump instruction then directs the processor to start executing code in the BIOS. If the system has just been powered up or the reset button was pressed ("cold boot"), the full power-on self-test (POST) is run. If Ctrl+Alt+Delete was initiated ("warm boot"), a special flag value is detected in Nonvolatile memory (NVRAM) and the BIOS does not run the POST. This saves the time otherwise used to detect and test all memory. The NVRAM is in the real-time clock (RTC).

The power-on self-test tests, identifies, and initializes system devices such as the CPU, RAM, interrupt and DMA controllers and other parts of the chipset, video display card, keyboard, hard disk drive, optical disc drive and other basic hardware. The BIOS then locates boot loader software held on a storage device designated as a 'boot device', such as a hard disk, a floppy disk, CD, or DVD, and loads and executes that software, giving it control of the PC.[8] This process is known as *booting*, or booting up, which is short for "bootstrapping".

Boot devices

The BIOS selects candidate boot devices using information collected by POST and configuration information from EEPROM, CMOS RAM or, in the earliest PCs, DIP switches. Option ROMs may also influence or supplant the boot process defined by the motherboard BIOS ROM. The BIOS checks each device in order to see if it is bootable. For a disk drive or a device that logically emulates a disk drive, such as an USB Flash drive or perhaps a tape drive, to perform this check the BIOS attempts to load the first sector (boot sector) from the disk to memory address 0x007C00, and checks for the boot sector signature 0x55 0xAA in the last two bytes of the (512 byte long) sector. If the sector cannot be read (due to a missing or blank disk, or due to a hardware failure), or if the sector does not end with the boot signature, the BIOS considers the disk unbootable and proceeds to check the next device. Another device such as a network adapter attempts booting by a procedure that is defined by its option ROM (or the equivalent integrated into the motherboard BIOS ROM). The BIOS proceeds to test each device sequentially until a bootable device is found, at which time the BIOS transfers control to the loaded sector with a jump instruction to its first byte at address 0x007C00 (1 KiB below the 32 KiB mark).

The behavior if the BIOS doesn't find a bootable device has varied as personal computers developed. The original IBM PC and XT had Microsoft Cassette BASIC in ROM, and if no bootable device was found, ROM BASIC was started. Therefore, barring a hardware failure, an original IBM PC or XT would never fail to boot, either into BASIC or from disk. One model of PC was available with no disk drive; a cassette recorder could be attached via the cassette port on the rear, for loading and saving BASIC programs to tape. Since few programs used BASIC in ROM, clone PC makers left it out; a computer that failed to boot from a disk would display "No ROM BASIC" and halt. Later computers would display a message like "No bootable disk

found." Modern BIOSes may display nothing or may automatically enter the BIOS configuration utility when the boot process fails.

Historically, the BIOS would try to boot from a floppy drive first and a hard disk second. CD or DVD booting is an extension of this. With the El Torito optical media boot standard, the optical drive actually emulates a 3.5" high-density floppy disk to the BIOS for boot purposes. Optical disks are a special case, because their lowest level of data organization is typically a fairly high-level file system (e.g. ISO 9660 for CD-ROM). To read the "first sector" of a CD-ROM or DVD-ROM is not a simply defined operation. The complexity of the medium makes it difficult to write a useful boot program in one sector. Therefore, optical media booting uses the El Torito standard, which specifies a way for an optical disk to contain an image of a high-density ("1.44MB") floppy disk and for the drive to provide access to this disk image in a simple manner that emulates floppy disk drive operations. Therefore, CD-ROM drives boot as emulated floppy disk drives; the bootable virtual floppy disk can then contain software that provides access to the optical medium in its native format.

A little-known feature of the original IBM BIOS versions is that before beginning the normal boot process they would attempt to load a program through the keyboard port. This was intended for factory test or diagnostic purposes. This was of limited utility outside of factory or repair facilities.[9]

BIOS extensions

In the IBM PC and AT, peripheral cards such as hard-drive controllers and video display adapters had their own BIOS extension option ROMs, which provided additional functionality. Code in these extensions runs before the operating system is loaded from mass storage. These ROMs can test and initialize hardware, add BIOS services, or replace BIOS services with their own versions of those services. For example, a SCSI controller usually has a BIOS extension ROM that adds support for hard drives connected through that controller. Some video cards have extension ROMs that replace the video services of the motherboard BIOS with their own video services. BIOS extension ROMs gain total control of the machine, so they may never return control to the BIOS that invoked them. An extension ROM could in principle contain an entire operating system or an application program, or it could implement an entirely different boot process such as booting from a network. Operation of an IBM-compatible computer system can be completely changed by removing or inserting an adapter card (or a ROM chip).

A computer system can contain several BIOS firmware chips. The motherboard BIOS typically contains code to access hardware components necessary for bootstrapping the system, such as the keyboard, display, and storage. In addition, plug-in adapter cards such as SCSI, RAID, network interface cards, and video boards often include their own BIOS (e.g. Video BIOS), complementing or replacing the system BIOS code for the given component. Even devices built into the motherboard can behave in this way; their option ROMs can be stored as separate code on the main BIOS flash chip, and upgraded either in tandem with, or separately from, the main BIOS.

An add-in card requires an option ROM if it needs to be used before the operating system can be loaded (usually this means it is required in the bootstrapping process), and is not supported by

the main BIOS.

After completing the POST, the motherboard BIOS scans for extension ROMs in an area of the "upper memory area" space and runs each ROM found, in order. To discover memory-mapped ISA option ROMs during the boot process, BIOS implementations scan real-mode address space from 0x0C0000 to 0x0F0000 on 2 KiB boundaries, looking for a ROM *signature*: 0x55 followed by 0xAA. In a valid expansion ROM, this signature is followed by a single byte indicating the number of 512-byte blocks it occupies in real memory. The next byte contains an offset describing the option ROM's entry point. A checksum of the specified number of 512-byte blocks is calculated, and if the ROM has a valid checksum the BIOS transfers control to the specified entry address. At this point, the expansion ROM code takes over, using BIOS services to register interrupt vectors for use by post-boot applications, to provide a user configuration interface, or to display diagnostic information.

There are many methods and utilities for examining the contents of various motherboard BIOS and expansion ROMs, such as Microsoft DEBUG or the Unix dd.

Boot environment

The environment for the boot program is very simple: the CPU is in real mode and the general-purpose and segment registers are undefined. All BIOS services are available, and the memory below address 0x000500 contains the interrupt vector table and the 256-byte BIOS data area, but the boot program must set up its own stack (or at least MS-DOS 6 acts like it must). All memory at and above address 0x000500 can be used by the boot program; it may even overwrite itself. The BIOS initializes a reserved block of system RAM with various parameters initialized during the POST. The interrupt vectors corresponding to the BIOS interrupts have been set to point at the appropriate entry points in the BIOS.

Operating system services

The BIOS ROM is customized to the particular manufacturer's hardware, allowing low-level services (such as reading a keystroke or writing a sector of data to diskette) to be provided in a standardized way to an operating system. For example, an IBM PC might have either a monochrome or a color display adapter (using different display memory addresses and hardware), but a single, standard, BIOS system call may be invoked to display a character at a specified position on the screen in text mode.

The BIOS provides a small library of basic input/output functions to operate peripherals (such as the keyboard, rudimentary text and graphics display functions and so forth). When using MS-DOS, BIOS services could be accessed by an application program (or by MS-DOS) by executing an INT 13H interrupt instruction to access disk sectors, or one of a number of other documented BIOS interrupt calls to access video display, keyboard, cassette, and other devices.

Operating systems and executive software, designed to supersede this basic firmware functionality, provide replacement software interfaces to applications. This began even in the 1980s under MS-DOS, when programmers observed that using the BIOS video services for graphics display was very slow. To increase the speed of screen output, many programs

bypassed the BIOS and programmed the video display hardware directly. Since the AT-compatible BIOS ran in Intel real mode, operating systems on '286 and later processors required hardware device drivers compatible with protected mode operation to replace BIOS services. In modern personal computers the BIOS is used only during booting and initial loading of system software. Before the operating system's first graphical screen is displayed, input and output are typically handled through BIOS. A boot menu such as the textual menu of Windows that allows one to choose an operating system to boot or to boot into Safe Mode or to use the last known good configuration, is displayed and receives keyboard input through BIOS.

BIOS components

In Intel systems, the BIOS may contain components such as the Memory Reference Code (MRC), which is responsible for handling memory timings and related hardware settings.[10]:8[11]

Setup utility

Historically, the BIOS in the IBM PC and XT had no built-in user interface. The BIOS versions in earlier PCs (XT-class) were not software configurable; instead, users set the options via DIP switches on the motherboard. Later computers, including all IBM-compatibles with 80286 CPUs, had a battery-backed nonvolatile BIOS memory (CMOS RAM chip) that held BIOS settings.[12] These settings, such as video-adapter type, memory size, and hard-disk parameters, could only be configured by running a configuration program from a disk, not built into the ROM. A special "reference diskette" was inserted in an IBM AT to configure settings such as memory size.

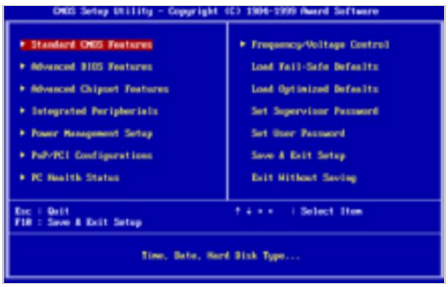
Early BIOS versions did not have passwords or boot-device selection options. The BIOS was hard-coded to boot from the first floppy drive, or, if that failed, the first hard disk. Access control in early AT-class machines was by a physical keylock switch (which was not hard to defeat if the computer case could be opened). Anyone who could switch on the computer could boot it.[*citation needed*]

Later, 386-class computers started integrating the BIOS setup utility in the ROM itself, alongside with the BIOS code; the computers usually booted into the BIOS setup utility if a certain key or key combination is pressed, otherwise the actual BIOS code was booted.

A modern BIOS setup utility has a menu-based user interface (UI) accessed by pressing a certain key on the keyboard when the PC starts. Usually the key is advertised for short time during the early startup, for example "Press F1 to enter CMOS setup". The actual key depends on specific hardware. In the BIOS setup utility, a user can:

- configure hardware
- set the system clock
- enable or disable system components
- select which devices are potential boot devices
- set various password prompts, such as a password for securing access to the BIOS user

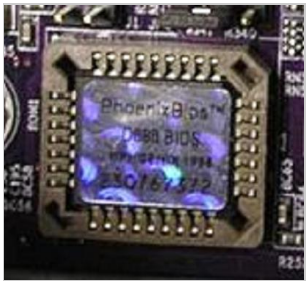
interface functions itself and preventing malicious users from booting the system from unauthorized peripheral devices



Award BIOS setup utility on a standard PC

Chips

The original IBM PC BIOS (and cassette BASIC) was stored on mask-programmed read-only memory (ROM) chips in sockets on the motherboard. ROMs could be replaced, but not altered, by users. To allow for updates, many compatible computers used re-programmable memory devices such as EEPROM and later flash memory devices. According to Robert Braver, the president of the BIOS manufacturer Micro Firmware, **Flash BIOS** chips became common around 1995 because the electrically erasable PROM (EEPROM) chips are cheaper and easier to program than standard ultraviolet erasable PROM (EPROM) chips. Flash chips are programmed (and re-programmed) in-circuit, while EPROM chips need to be removed from the motherboard for re-programming.[13] BIOS versions are upgraded to take advantage of newer versions of hardware and to correct bugs in previous revisions of BIOSes.[14]



PhoenixBIOS D686. This BIOS chip is housed in a PLCC package in a socket.

Beginning with the IBM AT, PCs supported a hardware clock settable through BIOS. It had a century bit which allowed for manually changing the century when the year 2000 happened. Most BIOS revisions created in 1995 and nearly all BIOS revisions in 1997 supported the year 2000 by setting the century bit automatically when the clock rolled past midnight, December 31, 1999.[15]

The first flash chips were attached to the ISA bus. Starting in 1997, the BIOS flash moved to the LPC bus, a functional replacement for ISA, following a new standard implementation known as "firmware hub" (FWH). In 2006, the first systems supporting a Serial Peripheral Interface (SPI) appeared, and the BIOS flash memory moved again.[*citation needed*]

The size of the BIOS, and the capacities of the ROM, EEPROM and other media it may be stored on, has increased over time as new features have been added to the code; BIOS versions now exist with sizes up to 16 megabytes. Some modern motherboards are including even bigger NAND flash memory ICs on board which are capable of storing whole compact operating systems, such as some Linux distributions. For example, some ASUS motherboards included SplashTop Linux embedded into their NAND flash memory ICs.[16]

Another type of firmware chip was found on the IBM PC and early compatibles. In the PC and AT, the keyboard interface was controlled by a microcontroller with its own programmable memory. On the IBM AT, this was a 40 pin socketed device. Some manufacturers used an EPROM version of this chip which resembled an EPROM. In the AT, this controller was also assigned the A20 gate function to manage memory above the 1 megabyte range; occasionally an upgrade of this "keyboard BIOS" was necessary to take advantage of software that could use

upper memory.[*citation needed*]

Flashing the BIOS

In modern PCs the BIOS is stored in rewritable memory, allowing the contents to be replaced or 'rewritten'. This rewriting of the contents is sometimes termed **flashing**. This can be done by a special program, usually provided by the system's manufacturer, or at POST, with a BIOS image in a hard drive or USB flash drive. A file containing such contents is sometimes termed 'a BIOS image'. A BIOS might be reflashed in order to upgrade to a newer version to fix bugs or provide improved performance or to support newer hardware, or a reflashing operation might be needed to fix a damaged BIOS.

Overclocking

Some BIOS chips allow overclocking, an action in which the CPU is adjusted to a higher clock rate than its factory preset. Overclocking may, however, seriously compromise system reliability in insufficiently cooled computers and generally shorten component lifespan. Overclocking, incorrectly performed, may also cause components to overheat so quickly that they destroy themselves.

BIOS chip vulnerabilities

An American Megatrends BIOS showing a “Intel CPU uCode Loading Error” after a failed attempt to upload microcode patches into the CPU.

EEPROM chips are advantageous because they could be easily updated by the user; hardware manufacturers frequently issue BIOS updates to upgrade their products, improve compatibility and remove bugs. However, this advantage had the risk that an improperly executed or aborted BIOS update could render the computer or device unusable. To avoid these situations, more recent BIOSes use a "boot block"; a portion of the BIOS which runs first and must be updated separately. This code verifies if the rest of the BIOS is intact (using hash checksums or other methods) before transferring control to it. If the boot block detects any corruption in the main BIOS, it will typically warn the user that a recovery process must be initiated by booting from removable media (floppy, CD or USB memory) so the user can try flashing the BIOS again. Some motherboards have a *backup* BIOS (sometimes referred to as DualBIOS boards) to recover from BIOS corruptions.

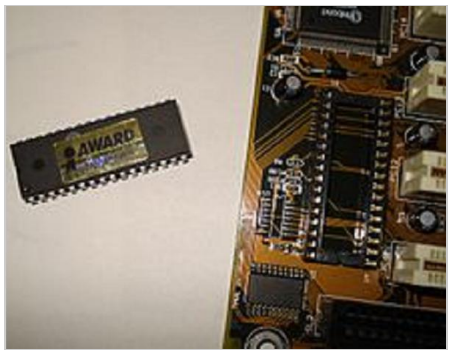
Virus attacks

There are at least four known BIOS attack viruses, two of which were for demonstration purposes. The first one found in the wild was Mebromi, targeting Chinese users.

Main article: CIH (computer virus)

The first BIOS virus was CIH, whose name matches the initials of its creator, Chen Ing Hau. CIH was also called the "Chernobyl Virus", because its payload date was 1999-04-26, the 13th anniversary of the Chernobyl accident.

CIH appeared in mid-1998 and became active in April 1999. It was able to erase flash ROM BIOS content. Often, infected computers could no longer boot, and people had to remove the flash ROM IC from the motherboard and reprogram it. CIH targeted the then-widespread Intel i430TX motherboard chipset and took advantage of the fact that the Windows 9x operating systems, also widespread at the time, allowed direct hardware access to all programs.



Detached BIOS Chip

Modern systems are not vulnerable to CIH because of a variety of chipsets being used which are incompatible with the Intel i430TX chipset, and also other flash ROM IC types. There is also extra protection from accidental BIOS rewrites in the form of boot blocks which are protected from accidental overwrite or dual and quad BIOS equipped systems which may, in the event of a crash, use a backup BIOS. Also, all modern operating systems such as FreeBSD, Linux, OS X, Windows NT-based Windows OS like Windows 2000, Windows XP and newer, do not allow user-mode programs to have direct hardware access. As a result, as of 2008, CIH has become essentially harmless, at worst causing annoyance by infecting executable files and from antivirus software. Other BIOS viruses remain possible, however;[17] since most Windows home users without Windows Vista/7's UAC run all applications with administrative privileges, a modern CIH-like virus could in principle still gain access to hardware without first using an exploit. The operating system OpenBSD prevents all users from having this access and the grsecurity patch for the linux kernel also prevents this direct hardware access by default, the difference being an attacker requiring a much more difficult kernel level exploit or reboot of the machine.

The second BIOS virus was a technique presented by John Heasman, principal security consultant for UK-based Next-Generation Security Software. In 2006, at the Black Hat Security Conference, he showed how to elevate privileges and read physical memory, using malicious procedures that replaced normal ACPI functions stored in flash memory.

The third BIOS virus was a technique called "Persistent BIOS infection." It appeared in 2009 at the CanSecWest Security Conference in Vancouver, and at the SyScan Security Conference in Singapore. Researchers Anibal Sacco[18] and Alfredo Ortega, from Core Security Technologies, demonstrated how to insert malicious code into the decompression routines in the BIOS, allowing for nearly full control of the PC at start-up, even before the operating system is booted.

The proof-of-concept does not exploit a flaw in the BIOS implementation, but only involves the normal BIOS flashing procedures. Thus, it requires physical access to the machine, or for the user to be root. Despite these requirements, Ortega underlined the profound implications of his and Sacco's discovery: “We can patch a driver to drop a fully working rootkit. We even have a little code that can remove or disable antivirus.”[19]

Mebromi is a trojan which targets computers with AwardBIOS, Microsoft Windows, and antivirus software from two Chinese companies: Rising Antivirus and Jiangmin KV Antivirus.[20] [21] [22] Mebromi installs a rootkit which infects the master boot record.

In a December 2013 interview with CBS 60 Minutes, Deborah Plunkett, Information Assurance Director for the US National Security Agency claimed that NSA analysts had uncovered and thwarted a possible BIOS attack by a foreign nation state. The attack on the world's computers could have allegedly "literally taken down the US economy." The segment further cites anonymous cyber security experts briefed on the operation as alleging the plot was conceived in China.[23] A later article in The Guardian cast doubt on the likelihood of such a threat, quoting Berkeley computer-science researcher Nicholas Weaver, Matt Blaze, a computer and information sciences professor at the University of Pennsylvania, and cybersecurity expert Robert David Graham in an analysis of the NSA's claims.[24]

BIOS Boot Specification

If the expansion ROM wishes to change the way the system boots (such as from a network device or a SCSI adapter for which the BIOS has no driver code), it can use the *BIOS Boot Specification* (BBS) API to register its ability to do so. Once the expansion ROMs have registered using the BBS APIs, the user can select among the available boot options from within the BIOS's user interface. This is why most BBS compliant PC BIOS implementations will not allow the user to enter the BIOS's user interface until the expansion ROMs have finished executing and registering themselves with the BBS API.[*citation needed*] The specification can be downloaded from the ACPICA website. The official title is BIOS Boot Specification (Version 1.01, 11 January 1996) and is available here: .[25]

Changing role of the BIOS

BIOS services are not used by modern multitasking GUI operating systems after they initially load, so the importance of the primary part of BIOS is greatly reduced from what it was initially. The limitations of the AT-compatible BIOS were its 16-bit processor mode, 1 MByte addressable space and reliance on PC AT hardware.

Some operating systems, for example MS-DOS, rely on the BIOS to carry out most input/output tasks within the PC.[26] Because the BIOS still runs in 16-bit real mode, calling BIOS services directly is inefficient for protected-mode operating systems. A number of larger, more powerful servers and workstations use a platform-independent Open Firmware (IEEE-1275) based on the Forth programming language; it is included with Sun's SPARC computers, IBM's RS/6000 line, and other PowerPC systems such as the CHRP motherboards. Later x86-based personal computer operating systems, like Windows NT, use their own, native drivers; this makes it much easier to extend support to new hardware.

Later BIOS took on more complex functions, by way of interfaces such as ACPI; these functions include power management, hot swapping, thermal management.

As of 2011, the BIOS is being replaced by the more complex Extensible Firmware Interface (EFI) in many new machines. EFI is a specification which replaces the runtime interface of the legacy BIOS. Initially written for the Itanium architecture, EFI is now available for x86 and x86-64 platforms; the specification development is driven by The Unified EFI Forum, an industry Special Interest Group. EFI booting has been supported in only Microsoft Windows versions supporting GPT,[27] the Linux kernel 2.6.1 and later, and Mac OS X on Intel-based Macs.[28] However, the distinction between BIOS and EFI is rarely made in terminology by the average computer user, making BIOS a catch-all term for both systems.

SLIC

Some BIOSes contain a "SLIC" (software licensing description table), a digital signature placed inside the BIOS by the manufacturer, for example Dell. (It is often casually called a BIOS tattoo or a tattooed BIOS.) This SLIC is inserted in the ACPI table and contains no active code.

Computer manufacturers that distribute OEM versions of Microsoft Windows and Microsoft application software can use the SLIC to authenticate licensing to the OEM Windows Installation disk and system recovery disc containing Windows software. Systems having a SLIC can be preactivated with an OEM product key, and they verify an XML formatted OEM certificate against the SLIC in the BIOS as a means of self-activating (see System Locked Preinstallation). If a user performs a fresh install of Windows, they will need to have possession of both the OEM key and the digital certificate for their SLIC in order to bypass activation; in practice this is extremely unlikely and hence the only real way this can be achieved is if the user performs a restore using a pre-customised image provided by the OEM. Cracks for non-genuine Windows distributions usually edit the SLIC or emulate it in order to bypass Windows activation.

Reprogrammable microcode

Intel processors have reprogrammable microcode since the P6 microarchitecture.[29][30] The BIOS may contain patches to the processor code to allow errors in the initial processor code to be fixed, updating the processor microcode each time the system is powered up. Otherwise, an expensive processor swap would be required.[31] For example, the Pentium FDIV bug became an expensive fiasco for Intel that required a product recall because the original Pentium did not have patchable microcode.

The BIOS business

IBM published the entire listings of the BIOS for its original PC, PC XT, PC AT, and other contemporary PC models, in an appendix of the Technical Reference manual for each machine type. The effect of the publication of the BIOS listings is that anyone can see exactly what a definitive BIOS does and how it does it. Phoenix Technology was the first company to write a fully compatible and completely legal BIOS through clean-room reverse engineering.

New standards grafted onto the BIOS are usually without complete public documentation or any

BIOS listings. As a result, it is not as easy to learn the intimate details about the many non-IBM additions to BIOS as about the core BIOS services.

Most PC motherboard suppliers license a BIOS "core" and toolkit from a commercial third-party, known as an "independent BIOS vendor" or IBV. The motherboard manufacturer then customizes this BIOS to suit its own hardware. For this reason, updated BIOSes are normally obtained directly from the motherboard manufacturer. Major BIOS vendors include American Megatrends (AMI), Insyde Software, Phoenix Technologies and Byosoft. Former vendors include Award Software and Microid Research which were acquired by Phoenix Technologies in 1998; Phoenix later phased out the Award Brand name. General Software, which was also acquired by Phoenix in 2007, sold BIOS for Intel processor based embedded systems.

The open source community increased their effort to develop a replacement for proprietary BIOSes and their future incarnations with an open sourced counterpart through the coreboot and OpenBIOS/Open Firmware projects. AMD provided product specifications for some chipsets, and Google is sponsoring the project. Motherboard manufacturer Tyan offers coreboot next to the standard BIOS with their Opteron line of motherboards. MSI and Gigabyte Technology have followed suit with the MSI K9ND MS-9282 and MSI K9SD MS-9185 resp. the M57SLI-S4 models.

Comparison

Comparison of different BIOS implementations

	AwardBIOS	AMIBIOS	Insyde	SeaBIOS
License	Proprietary	Proprietary	Proprietary	LGPL v3
Maintained / developed	No	Yes	Yes	Yes
32-bit PCI BIOS calls	?	?	?	Yes
AHCI	Yes	Yes	Yes	Yes
APM	Yes	Yes	Yes (1.2)	Yes (1.2)
BBS	Yes	Yes	Yes	Yes
Boot menu	Yes	Yes	Yes	Yes
Compression	Yes (LHA)	Yes (LHA)	Yes (RLE)	Yes (LZMA)
CMOS	Yes	Yes	Yes	Yes
EDD	Yes	Yes	Yes	Yes (3.0)
ESCD	Yes	Yes	?	No
Flash from ROM	?	Yes	?	No
Language	Assembly	Assembly	Assembly	C
LBA	Yes (48)	Yes (48)	Yes	Yes (48)
MultiProcessor Specification	Yes	Yes	Yes	Yes
Option ROM	Yes	Yes	Yes	Yes
Password	Yes	Yes	Yes	No
PMM	?	Yes	?	Yes
Setup screen	Yes	Yes	Yes	No
SMBIOS	Yes	Yes	Yes	Yes (2.4)
Splash screen	Yes	Yes (PCX)	Yes	Yes (BMP, JPG)
USB booting	Yes	Yes	Yes	Yes
USB hub	?	?	?	Yes
USB keyboard	Yes	Yes	Yes	Yes
USB mouse	Yes	Yes	Yes	Yes

See also

- Advanced Configuration and Power Interface (ACPI)
- Advanced Host Controller Interface (AHCI)
- ARCS
- coreboot, a project whose aim is to create a free and open source replacement for the BIOS
- e820h memory map
- Extended System Configuration Data

- Double boot
- Memtest86
- Open Firmware
- Plug and play
- System Management BIOS
- VESA BIOS Extensions
- Video BIOS
- XDK Debug BIOS for Xbox video game console
- Year 2000 problem

References

1. ^ The PC Guide - System BIOS (http://www.pcguide.com/ref/mbsys/bios/index.htm)
2. ^ ***a b*** Kildall, Gary A. (June 1975), *CP/M 1.1 or 1.2 BIOS and BDOS for Lawrence Livermore Laboratories*, "An excerpt of the BDOS.PLM file header in the PL/M source code of CP/M 1.1 or CP/M 1.2 for Lawrence Livermore Laboratories (LLL):

```
[...]
/* C P / M  B A S I C  I / O  S Y S T E M  (B I O S)
           COPYRIGHT (C) GARY A. KILDALL
                JUNE, 1975               */
[...]
/* B A S I C  D I S K  O P E R A T I N G  S Y S T E M  (B D O S)
           COPYRIGHT (C) GARY A. KILDALL
                JUNE, 1975                  */
[...]
```

   "
3. ^ ***a b c*** Kildall, Gary A. (January 1980). "The History of CP/M, THE EVOLUTION OF AN INDUSTRY: ONE PERSON'S VIEWPOINT" (http://www.retrotechnology.com /dri/CPM_history_kildall.txt) (Vol. 5, No. 1, Number 41 ed.). Dr. Dobb's Journal of Computer Calisthenics & Orthodontia. pp. 6–7. Retrieved 2013-06-03. "[...] The first commercial licensing of CP/M took place in 1975 with contracts between Digital Systems and Omron of America for use in their intelligent terminal, and with Lawrence Livermore Laboratories where CP/M was used to monitor programs in the Octopus network. Little attention was paid to CP/M for about a year. In my spare time, I worked to improve overall facilities [...] By this time, CP/M had been adapted for four different controllers. [...] In 1976, Glenn Ewing approached me with a problem: Imsai, Incorporated, for whom Glenn consulted, had shipped a large number of disk subsystems with a promise that an operating system would follow. I was somewhat reluctant to adapt CP/M to yet another controller, and thus the notion of a separated Basic I/O System (BIOS) evolved. In principle, the hardware dependent portions of CP/M were concentrated in the BIOS, thus allowing Glenn, or anyone else, to adapt CP/M to the Imsai equipment. Imsai was subsequently licensed to distribute CP/M version 1.3 which eventually evolved into an operating system called IMDOS. [...]"
4. ^ Bradley, Tony. "R.I.P. Bios: A Uefi Primer" (http://www.pcworld.com/article/248426 /r_i_p_bios_a_uefi_primer.html). PCWorld. Retrieved 2014-01-27.
5. ^ Swaine, Michael (1997-04-01). "Gary Kildall and Collegial Entrepreneurship" (http://www.ddj.com/184410428). *Dr. Dobb's Journal*. Retrieved 2006-11-20.

6. ^ Killian, A. Joseph "Joe" (2001). "Gary Kildall's CP/M: Some early CP/M history - 1976-1977" (http://www.imsai.net/history/imsai_history/cp-m_history.htm). Thomas "Todd" Fischer, IMSAI. Retrieved 2013-06-03. "[...] When we failed to produce an operating system in a timely manner, Glenn started talking with Gary about CPM, which Gary had written for Intel under contract. It took several months of twisting Gary's arm to get Gary to port it to the 8080. The final success came when Glenn talked Gary into just separating the I/O from the rest of it, with Glenn promising to re-write the I/O module for the IMSAI 8080 (which he did). So CPM on the IMSAI was a joint effort between Glenn and Gary. [...]"
7. ^ Fraley, Bob; Spicer, Dag (2007-01-26). "Oral History of Joseph Killian, Interviewed by: Bob Fraley, Edited by: Dag Spicer, Recorded: January 26, 2007, Mountain View, California, CHM Reference number: X3879.2007," (http://archive.computerhistory.org/resources/access/text/2012/10/102658016-05-01-acc.pdf). Computer History Museum. Retrieved 2013-06-03. "Killian: "[...] Intel had hired him a few months earlier to write a control program monitor to run on their little demo system for 8008 and now 8080. [...] Glenn knew this and he would be talking with Gary, and he started twisting Gary's arm. He said, "Hey Gary, why can't we run this in this IMSAI?" "The I/O's all different, won't run." But Glenn persists and finally makes a deal with Gary. He says, "Okay Gary, if you split out the I/O, I'll write the BIOS, basic I/O's system," and Glenn named it then. "We'll split it out separately. I'll write that part, as long as you can make a division in the program there." And he got Gary to do that and Glenn put those two pieces together and was running Gary's CP/M on an IMSAI. Glenn let us know that, and it wasn't too much later than Bill was down there making arrangements with Gary Kildall to license CP/M. [...] Now that the BIOS is separated out, anybody could write a BIOS for their machine, if it was 8080-based, and run this, so he started selling that separately under the company Digital Research that he formed and did quite well.""
8. ^ How StuffWorks: What BIOS Does (http://computer.howstuffworks.com/bios1.htm).
9. ^ page 5-27 *IBM Personal Computer Hardware Reference Library Technical Reference*, 1984, publication number 6361459
10. ^ Posted by Alex Watson, possibly repost from original content on custompc.com [unclear]. "The life and times of the modern motherboard" (http://www.bit-tech.net/custompc/features/601716/the-life-and-times-of-the-modern-motherboard/page1.html). *2007-11-27*. Retrieved 2 February 2013.
11. ^ David Hilber, Jr. (August 2009). "Considerations for Designing an Embedded Intel Architecture System with System Memory Down ®" (http://download.intel.com/embedded/processor/whitepaper/322506.pdf). Intel. Retrieved 2 February 2013.
12. ^ Torres, Gabriel (24 November 2004). "Introduction and Lithium Battery" (http://www.hardwaresecrets.com/article/81). *Replacing the Motherboard Battery*. hardwaresecrets.com. Retrieved June 20, 2013.
13. ^ "Decoding RAM & ROM (http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1997%2Fjun97%2F060997%2F060997.asp)." *Smart Computing*. June 1997. Volume 8, Issue 6.
14. ^ "Upgrading Your Flash BIOS For Plug And Play (http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1996%2Fmar96%2F96n0324%2F96n0324.asp)." *Smart Computing*. March 1996. Volume 7, Issue 3.
15. ^ "Time To Check BIOS (http://www.smartcomputing.com/editorial/article.asp?article=articles/archive/g0704/41u6/41u6.asp&guid=)." *Smart Computing*. April 1999. Volume 7, Issue 4.
16. ^ SplashTop's Instant-On Linux Desktop | Geek.com (http://www.geek.com/splashtops-instant-on-linux-desktop/)
17. ^ New BIOS Virus Withstands HDD Wipes (http://www.tomshardware.com/news/bios-virus-rootkit-security-backdoor,7400.html), March 27, 2009 by Marcus Yam - Tom's Hardware US
18. ^ Sacco, Anibal; Alfredo Ortéga. "Persistent BIOS Infection" (http://exploiting.wordpress.com/2009/03/23/cansecwest-was-great-here-the-presentation-slides/). *Exploiting Stuff*. Retrieved 2010-02-06.

19. ^ Fisher, Dennis. "Researchers unveil persistent BIOS attack methods" (http://threatpost.com/en_us/blogs/researchers-unveil-persistent-bios-attack-methods-031909). *Threat Post*. Archived (http://web.archive.org/web/20100130001722/http://threatpost.com/en_us/blogs/researchers-unveil-persistent-bios-attack-methods-031909) from the original on 30 January 2010. Retrieved 2010-02-06.
20. ^ Giuliani, Marco. "Mebromi: the first BIOS rootkit in the wild" (http://blog.webroot.com/2011/09/13/mebromi-the-first-bios-rootkit-in-the-wild/). *blog*. Retrieved 2011-09-19.
21. ^ "360发布"BMW病毒"技术分析报告" (http://bbs.360.cn/4005462/251096134.html). *blog*. Retrieved 2011-09-19.
22. ^ Yuan, Liang. "Trojan.Mebromi" (http://www.symantec.com/security_response/writeup.jsp?docid=2011-090609-4557-99). *Threat Response*. Retrieved 2011-09-19.
23. ^ "How did 60 Minutes get cameras into a spy agency?" (http://www.cbsnews.com/news/how-did-60-minutes-get-cameras-into-a-spy-agency/). CBS News. Retrieved 2014-04-15.
24. ^ Spencer Ackerman in Washington (2013-12-16). "NSA goes on 60 Minutes: the definitive facts behind CBS's flawed report | World news" (http://www.theguardian.com/world/2013/dec/16/nsa-surveillance-60-minutes-cbs-facts). theguardian.com. Retrieved 2014-01-27.
25. ^ *BIOS Boot Specification (Version 1.01, 11 January 1996)* (http://www.acpica.org/documentation/related_documents.php)
26. ^ Smart Computing Article - What Is The BIOS? (http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1994%2Fjuly94%2Fpcn0713%2Fpcn0713.asp) - Computing Basics July 1994 • Vol.5 Issue 7
27. ^ Windows and GPT FAQ (http://www.microsoft.com/whdc/device/storage/gpt_faq.mspx)
28. ^ Extensible Firmware Interface (EFI) and Unified EFI (UEFI) (http://www.intel.com/technology/efi/)
29. ^ Mueller, Scott (2001-06-08). "Processor Update Feature | Microprocessor Types and Specifications" (http://www.informit.com/articles/article.aspx?p=130978&seqNum=22). InformIT. Retrieved 2014-04-15.
30. ^ "Linux* Processor Microcode Data File" (https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=18148). *Download Center*. Downloadcenter.intel.com. 2009-09-23. Retrieved 2014-04-15.
31. ^ Scott Mueller, *Upgrading and repairing PCs 15th edition*, Que Publishing, 2003 ISBN 0-7897-2974-1, pages 109-110

Further reading

- *IBM Personal Computer Technical Reference* (Revised ed.). IBM Corporation. March 1983.
- *IBM Personal Computer AT Technical Reference*. IBM Personal Computer Hardware Reference Library. 0, 1, 2 (Revised ed.). IBM Corporation. March 1986 [1984-03]. 1502494, 6139362, 6183310, 6183312, 6183355, 6280070, 6280099.
- Phoenix Technologies, Ltd. (1989) [1987]. *System BIOS for IBM PC/XT/AT Computers and Compatibles — The Complete Guide to ROM-Based System Software*. Phoenix Technical Reference Series (1st ed.). Addison Wesley Publishing Company, Inc. ISBN 0-201-51806-6.
- Phoenix Technologies, Ltd. (1989) [1987]. *CBIOS for IBM PS/2 Computers and Compatibles — The Complete Guide to ROM-Based System Software for DOS*. Phoenix Technical Reference Series (1st ed.). Addison Wesley Publishing Company, Inc. ISBN 0-201-51804-X.

- Phoenix Technologies, Ltd. (1989) [1987]. *ABIOS for IBM PS/2 Computers and Compatibles — The Complete Guide to ROM-Based System Software for OS/2*. Phoenix Technical Reference Series (1st ed.). Addison Wesley Publishing Company, Inc. ISBN 0-201-51805-8.
- Ralf Brown's Interrupt List
- BIOS Disassembly Ninjutsu Uncovered, 1st edition (http://bioshacking.blogspot.com/2012/02/bios-disassembly-ninjutsu-uncovered-1st.html) (freely available book, PDF)

External links

- List of BIOS options (http://www.techarp.com/freebog.aspx?)
- How BIOS Works (http://computer.howstuffworks.com/bios.htm)
- Persistent BIOS Infection - Phrack #66 (http://www.phrack.com/archives/66/p66_0x07_Persistent%20BIOS%20infection_by_aLS%20and%20Alfredo.txt)

Specifications

- Preventing BIOS Failures Using Intel Boot Block Flash Memory (http://download.intel.com/design/flcomp/applnots/29219202.PDF) (December 1998)
- BIOS Boot Specification (http://www.phoenix.com/resources/specs-bbs101.pdf) 1.01 (January 1996)
- Implementing a Plug and Play BIOS Using Intel's Boot Block Flash Memory (http://download.intel.com/design/flcomp/support/applnots/29216101.pdf) (February 1995)