BIOS

For IBM PC compatible computers, **BIOS** (/ˈbaɪɒs/ *BY-oss*; an acronym for **Basic Input/Output System** and also known as the **System BIOS**, **ROM BIOS** or **PC BIOS**) is non-volatile firmware used to perform hardware initialization during the booting process (power-on startup), and to provide runtime services for operating systems and programs.[1] The BIOS firmware comes pre-installed on a personal computer's system board, and it is the first software run when powered on. The name originates from the Basic Input/Output System used in the CP/M operating system in 1975.[2][3] Originally proprietary to the IBM PC, the BIOS has been reverse engineered by companies looking to create compatible systems. The interface of that original system serves as a *de facto* standard.

A pair of AMD BIOS chips for a Dell 310 computer from the late 1980s

The BIOS in modern PCs initializes and tests the system hardware components, and loads a boot loader or an operating system from a mass memory device. In the era of MS-DOS, the BIOS provided a hardware abstraction layer for the keyboard, display, and other input/output (I/O) devices that standardized an interface to application programs and the operating system. More recent operating systems do not use the BIOS after loading, instead accessing the hardware components directly

Most BIOS implementations are specifically designed to work with a particular computer or motherboard model, by interfacing with various devices that make up the complementary system chipset. Originally, BIOS firmware was stored in a ROM chip on the PC motherboard. In modern computer systems, the BIOS contents are stored on flash memory so it can be rewritten without removing the chip from the motherboard. This allows easy, end-user updates to the BIOS firmware so new features can be added or bugs can be fixed, but it also creates a possibility for the computer to become infected with BIOS rootkits. Furthermore, a BIOS upgrade that fails can brick the motherboard permanently unless the system includes some form of backup for this case.

Unified Extensible Firmware Interface (UEFI) is a successor to BIOS, aiming to address its technical shortcomings.[4]

Contents

History

User interface

Operation
- System startup
- Boot process
  - Boot priority
  - Boot failure
- Boot environment

Extensions (option ROMs)
- Boot procedure
- Initialization
- Physical placement

Operating system services
- Processor microcode updates
- Identification
- Overclocking
- Modern use

Configuration
- Setup utility
- Reprogramming

Hardware

Vendors and products

Security

Alternatives and successors

See also

Notes

References

Further reading

External links

History

The term BIOS (Basic Input/Output System) was created by Gary Kildall[5] and first appeared in the CP/M operating system in 1975,[2][3][6][7][8] 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, PC DOS or DR-DOS contain a file called variously "IO.SYS", "IBMBIO.COM", "IBMBIO.SYS", or "DRBIOS.SYS"; this file is known as the "DOS BIOS" (also known as "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 analogue to the "CP/M BIOS".

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.

> ```
> /* 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 */
> ```
>
> — An excerpt from 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)[2]

User interface

The BIOS of the original IBM PC XT had no interactive user interface. Error codes or messages were displayed on the screen, or coded series of sounds were generated to signal errors when the power-on self-test (POST) had not proceeded to the point of successfully initializing a video display adapter. Options on the IBM PC and XT were set by switches and jumpers on the main board and on peripheral cards. Starting around the mid-1990s, it became typical for the BIOS ROM to include a *"BIOS configuration utility"* (BCU[9]) or "BIOS setup utility", accessed at system power-up by a particular key sequence. This program allowed the user to set system configuration options, of the type formerly set using DIP switches, through an interactive menu system controlled through the keyboard. In the interim period, IBM-compatible PCs—including the IBM AT—held configuration settings in battery-backed RAM and used a bootable configuration program on disk, not in the ROM, to set the configuration options contained in

> Starting:-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

this memory. The disk was supplied with the computer, and if it was lost the system settings could not be changed. The same applied in general to computers with an EISA bus, for which the configuration program was called an EISA Configuration Utility (ECU).

A modern Wintel-compatible computer provides a setup routine essentially unchanged in nature from the ROM-resident BIOS setup utilities of the late 1990s; the user can configure hardware options using the keyboard and video display. Also, when errors occur at boot time, a modern BIOS usually displays user-friendly error messages, often presented as pop-up boxes in a TUI style, and offers to enter the BIOS setup utility or to ignore the error and proceed if possible. Instead of battery-backed RAM, the modern Wintel machine may store the BIOS configuration settings in flash ROM, perhaps the same flash ROM that holds the BIOS itself.

Operation

System startup

Early Intel processors started at physical address 000FFFF0h. When a modern x86 microprocessor is reset, it starts in pseudo 16-bit real mode, initializing most registers to zero. The code segment register is initialized with selector F000h, base FFFF0000h, and limit FFFFh, so that execution starts at 4 GB minus 16 bytes (FFFFFFF0h).[10] The platform logic maps this address into the system ROM, mirroring address 000FFFF0h.

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 pressed ("warm boot"), a special flag value is stored in nonvolatile BIOS memory ("CMOS") before the processor is reset, and after the reset the BIOS startup code detects this flag and does not run the POST. This saves the time otherwise used to detect and test all memory

The POST checks, 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.

Early IBM PCs had a little-known routine in the POST that would attempt to download a maintenance program into RAM through the keyboard port before performing any other elements of the boot process, such as before scanning for option ROMs or executing a boot loader. (No serial or parallel ports were standard on early IBM PCs, but a keyboard port of either the XT or AT / PS/2 type has been standard on practically every PC and clone.) If the download was apparently successful, the BIOS would verify a checksum on it and then run it.[11][12] This feature was intended for factory test or diagnostic purposes; while it was of limited utility outside of factory or repair facilities, it could be used in a proprietary way to boot the PC as a satellite system to a host machine (as it was used in the manufacturing environment).

Boot process

> 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.
>
> — Gary Kildall[3]

> When we failed to produce an operating system in a timely manner, Glenn started talking with Gary about CPM ... 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

After the option ROM scan is completed and all detected ROM modules with valid checksums have been called, or immediately after POST in a BIOS version that does not scan for option ROMs, the BIOS calls INT 19h to start boot processing. Post-boot, programs loaded can also call INT 19h to reboot the system, but they must be careful to disable interrupts and other asynchronous hardware processes that may interfere with the BIOS rebooting process, or else the system may hang or crash while it is rebooting.

When INT 19h is called, the BIOS attempts to locate boot loader software held on a storage device designated as a "boot device", such as a hard disk, a floppy disk, CD, or DVD. It loads and executes the first boot software it finds, giving it control of the PC.[13] This is the process that is known as *booting* (sometimes informally called "booting up"), which is short for "bootstrapping".

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. Following the boot priority sequence in effect, 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 a 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 into RAM at memory address `0x0000:0x7C00`. If the sector cannot be read (due to a missing or unformatted disk, or due to a hardware failure), the BIOS considers the device unbootable and proceeds to check the next device. If the sector is read successfully, some BIOSes will also check for the boot sector signature 0x55 0xAA in the last two bytes of the sector (which is 512 bytes long), before accepting a boot sector and considering the device bootable.[nb 1]

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 `0x0000:0x7C00` (exactly 1 KiB below the 32 KiB mark); see MBR invocation and VBR invocation. (This location is one reason that an IBM PC requires at least 32 KiB of RAM in order to be equipped with a disk system; with 31 KiB or less, it would be impossible to boot from any disk, removable or fixed, using the BIOS boot protocol.) Most, but not all, BIOSes load the drive number (as used by INT 13h) of the boot drive into CPU register DL before jumping to the first byte of the loaded boot sector

Note well that the BIOS does not interpret or process the contents of the boot sector other than to possibly check for the boot sector signature in the last two bytes; all interpretation of data structures like MBR partition tables and so-called BIOS Parameter Blocks is done by the boot program in the boot sector itself or by other programs loaded through the boot process and is beyond the scope of BIOS. Nothing about BIOS predicates these data structures or impedes their replacement or improvement.

A non-disk 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. As such, option ROMs may also influence or supplant the boot process defined by the motherboard BIOS ROM.

Boot priority

8080 (which he did). So CPM on the IMSAI was a joint effort between Glenn and Gary.

— Joe Killian[7]

> “Glenn ... 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."”

— Joe Killian[8]

The user can control the boot process, to cause one medium to be booted instead of another when two or more bootable media are present, by taking advantage of the boot priority implemented by the BIOS. For example, most computers have a hard disk that is bootable, but usually there is a removable-media drive that has higher boot priority, so the user can cause a removable disk to be booted, simply by inserting it, without removing the hard disk drive or altering its contents to make it unbootable.

In most modern BIOSes, the boot priority order of all potentially bootable devices can be freely configured by the user through the BIOS configuration utility. In older BIOSes, limited boot priority options are selectable; in the earliest BIOSes, a fixed priority scheme was implemented, with floppy disk drives first, fixed disks (i.e. hard disks) second, and typically no other boot devices supported, subject to modification of these rules by installed option ROMs. The BIOS in an early PC also usually would only boot from the first floppy disk drive or the first hard disk drive, even if there were two drives of either type installed. All more advanced boot priority sequences evolved as incremental improvements on this basic system.

Historically the BIOS would try to boot from a floppy drive first and a hard disk second. The default for 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).

Reading the "first sector" of a CD-ROM or DVD-ROM is not a simply defined operation like it is on a floppy disk or a hard disk. Furthermore, the complexity of the medium makes it difficult to write a useful boot program in one sector, even though optical media sectors are typically 2048 bytes each, four times the standard 512-byte size of floppy and legacy hard disk sectors. 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 (1.44 MB) 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 contain software that provides access to the optical medium in its native format.

Boot failure

The behavior if the BIOS does not 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 by calling INT 18h. Therefore, barring a hardware failure, an original IBM PC or XT would never fail to boot, either into BASIC or from disk (or through an option ROM). One model of the original IBM 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; then a computer that failed to boot from a disk would display "No ROM BASIC" and halt (in response to INT 18h).

Later computers would display a message like "No bootable disk found"; some would prompt for a disk to be inserted and a key to be pressed, and when a key was pressed they would restart the boot process. A modern BIOS may display nothing or may automatically enter the BIOS configuration utility when the boot process fails. Unlike earlier BIOSes, modern versions are often written with the assumption that if the computer cannot be booted from a hard disk, the user will not have software that they want to boot from removable media instead. (Lately, typically it will only be a specialist computer technician who does that, only to get the computer back into a condition where it can be booted from the hard disk.)

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, except CS, SS, SP, and DL. CS is always zero and IP is initially `0x7C00`. Because boot programs are always loaded at this fixed address, there is no need or motivation for a boot program to be relocatable. DL contains the drive number, as used with INT 13h, of the boot device, unless the BIOS is one that does not set the drive number in DL – and then DL is undefined. SS:SP points to a valid stack that is presumably large enough to support hardware interrupts, but otherwise SS and SP are undefined. (A stack must be already set up in order for interrupts to be serviced, and interrupts must be enabled in order for the system timer-tick interrupt, which BIOS always uses at least to maintain the time-of-day count and which it initializes during POST, to be active and

for the keyboard to work. The keyboard works even if the BIOS keyboard service is not called; keystrokes are received and placed in the 15-character type-ahead buffer maintained by BIOS.) The boot program must set up its own stack (or at least MS-DOS 6 acts like it must), because the size of the stack set up by BIOS is unknown and its location is likewise variable; although the boot program can investigate the default stack by examining SS:SP, it is easier and shorter to just unconditionally set up a new stack.

At boot time, all BIOS services are available, and the memory below address `0x00400` contains the interrupt vector table. BIOS POST has initialized the system timers (8253 or 8254 IC), interrupt controller(s), DMA controller(s), and other motherboard/chipset hardware as necessary to bring all BIOS services to ready status. DRAM refresh for all system DRAM in conventional memory and extended memory, but not necessarily expanded memory, has been set up and is running. The interrupt vectors corresponding to the BIOS interrupts have been set to point at the appropriate entry points in the BIOS, hardware interrupt vectors for devices initialized by the BIOS have been set to point to the BIOS-provided ISRs, and some other interrupts, including ones that BIOS generates for programs to hook, have been set to a default dummy ISR that immediately returns. The BIOS maintains a reserved block of system RAM at addresses `0x00400–0x004FF` with various parameters initialized during the POST. All memory at and above address `0x00500` can be used by the boot program; it may even overwrite itself.

Extensions (option ROMs)

Peripheral cards such as some hard disk drive controllers and some video display adapters have their own BIOS extension option ROMs, which provide additional functionality to BIOS. Code in these extensions runs before the BIOS boots the system from mass storage. These ROMs typically test and initialize hardware, add new BIOS services, and augment or replace existing 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 can in fact do anything, and 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) that contains a BIOS extension ROM.

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 the card is not supported by the main BIOS and the card needs to be initialized or made accessible through BIOS services before the operating system can be loaded (usually this means it is required in the bootstrapping process). Even when it is not required, an option ROM can allow an adapter card to be used without loading driver software from a storage device after booting begins – with an option ROM, no time is taken to load the driver, the driver does not take up space in RAM nor on hard disk, and the driver software on the ROM always stays with the device so the two cannot be accidentally separated. Also, if the ROM is on the card, both the peripheral hardware and the driver software provided by the ROM are installed together with no extra effort to install the software. An additional advantage of ROM on some early PC systems (notably including the IBM PCjr) was that ROM was faster than main system RAM. (On modern systems, the case is very much the reverse of this, and BIOS ROM code is usually copied ("shadowed") into RAM so it will run faster.)

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 procedure

If an 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) in a cooperative way, 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. The specification can be downloaded from the ACPICA website. The official title is BIOS Boot Specification (Version 1.01, 11 January 1996).[14]

Also, if an expansion ROM wishes to change the way the system boots unilaterally, it can simply hook INT 19h or other interrupts normally called from interrupt 19h, such as INT 13h, the BIOS disk service, to intercept the BIOS boot process. Then it can replace the BIOS boot process with one of its own, or it can merely modify the boot sequence by inserting its own boot actions into it, by preventing the BIOS from detecting certain devices as bootable, or both. Before the BIOS Boot Specification was promulgated, this was the only way for expansion ROMs to implement boot capability for devices not supported for booting by the native BIOS of the motherboard.

Initialization

After the motherboard BIOS completes its POST, most BIOS versions search for option ROM modules, also called BIOS extension ROMs, and execute them. The motherboard BIOS scans for extension ROMs in a portion of the "upper memory area" (the part of the x86 real-mode address space at and above address 0xA0000) and runs each ROM found, in order. To discover memory-mapped ISA option ROMs, a BIOS implementation scans the real-mode address space from `0x0C0000` to `0x0F0000` on 2 KiB boundaries, looking for a two-byte 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 the expansion ROM occupies in real memory, and the next byte is the option ROM's entry point (also known as its "entry offset"). 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 entry address, which in a normal BIOS extension ROM should be the beginning of the extension's initialization routine.

At this point, the extension ROM code takes over, typically testing and initializing the hardware it controls and registering interrupt vectors for use by post-boot applications. It may use BIOS services (including those provided by previously initialized option ROMs) to provide a user configuration interface, to display diagnostic information, or to do anything else that it requires. While the actions mentioned are typical behaviors of BIOS extension ROMs, each option ROM receives total control of the computer and may do anything at all, as noted with more detail in the Extensions section below; it is possible that an option ROM will not return to BIOS, pre-empting the BIOS's boot sequence altogether.

An option ROM should normally return to the BIOS after completing its initialization process. Once (and if) an option ROM returns, the BIOS continues searching for more option ROMs, calling each as it is found, until the entire option ROM area in the memory space has been scanned.

Physical placement

Option ROMs normally reside on adapter cards. However, the original PC, and perhaps also the PC XT, have a spare ROM socket on the motherboard (the "system board" in IBM's terms) into which an option ROM can be inserted, and the four ROMs that contain the BASIC interpreter can also be removed and replaced with custom ROMs which can be option ROMs. The IBM PCjr is unique among PCs in having two ROM cartridge slots on the front. Cartridges in these slots map into the same region of the upper memory area used for option ROMs, and the cartridges can contain option ROM modules that the BIOS would recognize. The cartridges can also contain other types of ROM modules, such as BASIC programs, that are handled differently. One PCjr cartridge can contain several ROM modules of different types, possibly stored together in one ROM chip.

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 programs, including operating systems. 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 or graphics 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 functions, or by executing one of a number of other documented BIOS interrupt calls to access video display, keyboard, cassette, and other device functions.

Operating systems and executive software that are designed to supersede this basic firmware functionality provide replacement software interfaces to application software. Applications can also provide these services to themselves. 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. Other graphics programmers, particularly but not exclusively in the demoscene, observed that there were technical capabilities of the PC display adapters that were not supported by the IBM BIOS and could not be taken advantage of without circumventing it. Since the AT-compatible BIOS ran in Intel real mode, operating systems that ran in protected mode on 286 and later processors required hardware device drivers compatible with protected mode operation to replace BIOS services.

In modern personal computers running modern operating systems 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, which allows users to choose an operating system to boot, to boot into the safe mode, or to use the last known good configuration, is displayed through BIOS and receives keyboard input through BIOS.

Most modern PCs can still boot and run legacy operating systems such as MS-DOS or DR-DOS that rely heavily on BIOS for their console and disk I/O, providing that the system has a BIOS or BIOS-compatible firmware, which is not necessarily the case with UEFI-based PCs.

Processor microcode updates

Intel processors have reprogrammable microcode since the P6 microarchitecture.[15][16] The BIOS may contain patches to the processor microcode that fix errors in the initial processor microcode; reprogramming is not persistent, thus loading of microcode updates is performed each time the system is powered up. Without reprogrammable microcode, an expensive processor swap would be required;[17] for example, the Pentium FDIV bug became an expensive fiasco for Intel as it required a product recall because the original Pentium processor's defective microcode could not be reprogrammed.

Identification

Some BIOSes contain a software licensing description table (SLIC), a digital signature placed inside the BIOS by the original equipment manufacturer (OEM), for example Dell. The SLIC is inserted into the ACPI table and contains no active code.[18][19]

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 with an 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, SLP). If a user performs a fresh install of Windows, they will need to have possession of both the OEM key (either SLP or COA) and the digital certificate for their SLIC in order to bypass activation.[18] This can be achieved if the user performs a restore using a pre-customised image provided by the OEM. Power users can copy the necessary certificate files from the OEM image, decode the SLP product key, then perform SLP activation manually. Cracks for non-genuine Windows distributions usually edit the SLIC or emulate it in order to bypass Windows activation.

Overclocking

Some BIOS implementations allow overclocking, an action in which the CPU is adjusted to a higher clock rate than its manufacturer rating for guaranteed capability. Overclocking may, however, seriously compromise system reliability in insufficiently cooled computers and generally shorten component lifespan. Overclocking, when incorrectly performed, may also cause components to overheat so quickly that they mechanically destroy themselves.[20]

Modern use

Some operating systems, for example MS-DOS, rely on the BIOS to carry out most input/output tasks within the PC.[21]

Because the BIOS still runs in 16-bit real mode, calling BIOS services directly is inefficient for protected-mode operating systems. BIOS services are not used by modern multitasking operating systems after they initially load, so the importance of the primary part of BIOS is greatly reduced from what it was initially

Later BIOS implementations took on more complex functions, by including interfaces such as Advanced Configuration and Power Interface (ACPI); these functions include power management, hot swapping, and thermal management. At the same time, since 2010, BIOS technology is in a transitional process toward UEFI.[4]

Configuration

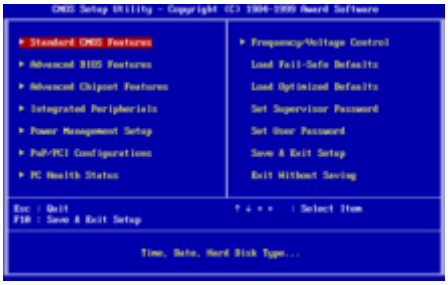
Setup utility

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.[22] 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.

Later, 386-class computers started integrating the BIOS setup utility in the ROM itself, alongside the BIOS code; these computers usually boot into the BIOS setup utility if a certain key or key combination is pressed, otherwise the BIOS POST and boot process are executed.

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. Features present in the BIOS setup utility typically include:



Award BIOS setup utility on a standard PC

- Configuring the hardware components, including setting their various operating modes and frequencies (for example, selecting how the storage controllers are visible to the operating system, or overclocking the CPU)
- Setting the system clock
- Enabling or disabling system components
- Selecting which devices are potential boot devices, and in which order booting from them will be attempted
- Setting various passwords, such as a password for securing access to the BIOS user interface functions itself and preventing malicious users from booting the system from unauthorized portable storage devices, a password for

booting the system, or a hard disk drive password that limits access to it and stays assigned even if the hard disk drive is moved to another computer

Reprogramming

In modern PCs the BIOS is stored in rewritable memory, allowing the contents to be replaced and modified. This rewriting of the contents is sometimes termed *flashing*, based on the common use of a kind of EEPROM known technically as "flash EEPROM" and colloquially as "flash memory". It 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.

BIOS replacement kit for a Dell 310 from the late 1980s. Included are two chips, a plastic holder for the chips, and a chip puller.

Hardware

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 EPROM 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.[23] BIOS versions are upgraded to take advantage of newer versions of hardware and to correct bugs in previous revisions of BIOSes.[24]

American MegatrendsBIOS 686. This BIOS chip is housed in aPLCC 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.[25]

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.

The size of the BIOS, and the capacity of the ROM, EEPROM, or 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. For contrast, the original IBM PC BIOS was contained in an 8 KiB mask ROM. 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.[26] However, the idea of including an operating system along with BIOS in the ROM of a PC is not new; in the 1980s, Microsoft offered a ROM option for MS-DOS, and it was included in the ROMs of some PC clones such as the Tandy 1000 HX.

Another type of firmware chip was found on the IBM PC AT and early compatibles. In the AT, the keyboard interface was controlled by a microcontroller with its own programmable memory. On the IBM AT, that was a 40-pin socketed device, while some manufacturers used an EPROM version of this chip which resembled an EPROM. This controller was also assigned the A20 gate

function to manage memory above the one-megabyte range; occasionally an upgrade of this "keyboard BIOS" was necessary to take advantage of software that could use upper memory

The BIOS may contain components such as the Memory Reference Code (MRC), which is responsible for handling memory timings and related hardware settings.[27]:8[28]

Vendors and products

Comparison of different BIOS implementations

Company	AwardBIOS	AMIBIOS	Insyde	SeaBIOS
License	Proprietary	Proprietary	Proprietary	LGPL v3
Maintained / developed	Yes	Yes	Yes	Yes
32-bit PCI BIOS calls	Yes	Yes	Yes	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[29])	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 (EPA)[30]	Yes (PCX)	Yes	Yes (BMP, JPG)
TPM	Unknown	Unknown	Unknown	Some
USB booting	Yes	Yes	Yes	Yes
USB hub	?	?	?	Yes
USB keyboard	Yes	Yes	Yes	Yes
USB mouse	Yes	Yes	Yes	Yes

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 *IBM PC 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.

In May 1984 Phoenix Software Associates released its first ROM-BIOS which enabled OEMs to build essentially 100%-compatible clones without having to reverse-engineer the IBM PC BIOS themselves, as Compaq had done for the Portable, helping fuel the growth in the PC compatibles industry and sales of non-IBM versions of DOS.[31] And the first American Megatrends (AMI) BIOS was released on 1986.

New standards grafted onto the BIOS are usually without complete public documentation or any BIOS listings. As a result, it is not a 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 that 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.

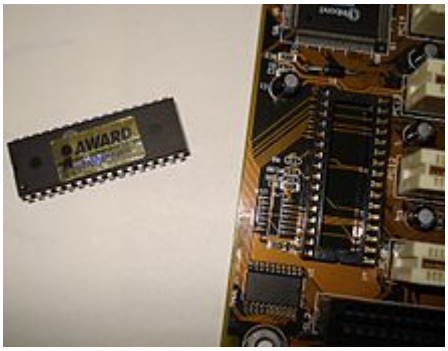
The open source community increased their effort to develop a replacement for proprietary BIOSes and their future incarnations with an open sourced counterpart through the libreboot, coreboot and OpenBIOS/Open Firmware projects. AMD provided product specifications for some chipsets, and Google is sponsoring 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.

Security

EEPROM chips are advantageous because they can be easily updated by the user; it is customary for hardware manufacturers to 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 flash drive) 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.

An American MegatrendsBIOS showing a "Intel CPU uCode Loading Error" after a failed attempt to upload microcode patches into the CPU

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.



A detached BIOS chip

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.

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, macOS, 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 triggering antivirus software. Other BIOS viruses remain possible, however;[32] 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[33] 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!"[34]

Mebromi is a trojan which targets computers with AwardBIOS, Microsoft Windows, and antivirus software from two Chinese companies: Rising Antivirus and Jiangmin KV Antivirus.[35][36][37] Mebromi installs a rootkit which infects the master boot record.

In a December 2013 interview with *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.[38] 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.[39]

Alternatives and successors

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 Intel 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,[40] the Linux kernel 2.6.1 and later, and macOS on Intel-based Macs.[41] As of 2014, new PC hardware predominantly ships with UEFI firmware. The architecture of the rootkit safeguard can also prevent the system from running the user's own software changes, which makes UEFI controversial as a BIOS replacement in the open hardware community.

Other alternatives to the functionality of the "Legacy BIOS" in the x86 world include coreboot and libreboot.

Some 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, along with the x86-based OLPC XO-1.

As of at least 2015, Apple has removed legacy BIOS support from MacBook Pro computers. As such the bless utility no longer supports the --legacy switch, and prints "Legacy mode not supported on this system". These Macs also cannot boot from CD-ROM or USB flash drives.

See also

- Unified Extensible Firmware Interface (UEFI)
- e820
- Extended System Configuration Data (ESCD)
- Double boot
- Legacy Plug and Play (PnP) – specifications supporting automated configuration of hardware devices, primarily those on the ISA bus
- Ralf Brown's Interrupt List (RBIL) – interrupts, calls, interfaces, data structures, memory and port addresses, and processor opcodes for the x86 architecture
- System Management BIOS (SMBIOS)
- VESA BIOS Extensions (VBE) – an interface for using compliant video boards at high resolutions and bit depths, beyond the standard BIOS support
- Fujitsu TBIOS (on Fujitsu FM Towns computers)

Notes

1. The signature at offset `+0x1FE` in boot sectors is `0x55 0xAA`, that is `0x55` at offset `+0x1FE` and `0xAA` at offset `+0x1FF`. Since little-endian representation must be assumed in the context of IBM PC compatible machines, this can be written as 16-bit word `0xAA55` in programs for x86 processors (note the swapped order), whereas it would have to be written as `0x55AA` in programs for other CPU architectures using a big-endian representation. Since this has been mixed up numerous times in books and even in original Microsoft reference documents, this article uses the offset-based byte-wise on-disk representation to avoid any possible misinterpretation.

References

1. "Ref — System BIOS" (http://www.pcguide.com/ref/mbsys/bios/index.htm). *PCGuide*. Archived (https://web.archive.org/web/20141221071646/http://www.pcguide.com/ref/mbsys/bios/index.htm) from the original on 21 December 2014. Retrieved 6 December 2014.
2. Kildall, Gary A. (June 1975), *CP/M 1.1 or 1.2 BIOS and BDOS for Lawrence Livermore Laboratories*
3. 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. Archived (https://web.archive.org/web/20161124221907/http://www.retrotechnology.com/dri/CPM_history_kildall.txt) from the original on 2016-11-24. Retrieved 2013-06-03.
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. Archived (https://web.archive.org/web/20140127090326/http://www.pcworld.com/article/248426/r_i_p_bios_a_uefi_primer.html) from the original on 2014-01-27. Retrieved 2014-01-27.
5. Swaine, Michael (1997-04-01). "Gary Kildall and Collegial Entrepreneurship" (http://www.ddj.com/184410428). *Dr. Dobb's Journal*. Archived (https://web.archive.org/web/20070124184442/http://www.ddj.com/184410428) from the original on 2007-01-24. Retrieved 2006-11-20.
6. Shustek, Len (2016-08-02). "In His Own Words: Gary Kildall" (http://www.computerhistory.org/atchm/in-his-own-words-gary-kildall/). *Remarkable People*. Computer History Museum. Archived (https://web.archive.org/web/20161217072842/http://www.computerhistory.org/atchm/in-his-own-words-gary-kildall/) from the original on 2016-12-17.
7. 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. Archived (https://web.archive.org/web/20121229064433/http://www.imsai.net/history/imsai_history/cp-m_history.htm) from the original on 2012-12-29. Retrieved 2013-06-03.

8. 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) (PDF). Computer History Museum. Archived (https://web.archive.org/web/20140714175258/http://archive.computerhistory.org/resources/access/text/2012/10/102658016-05-01-acc.pdf) (PDF) from the original on July 14, 2014. Retrieved 2013-06-03.
9. "HP BIOS Configuration Utility" (http://ftp.hp.com/pub/caps-softpaq/cmit/HP_BCU.html). Hewlett-Packard. 2013. Archived (https://web.archive.org/web/20150112201124/http://ftp.hp.com/pub/caps-softpaq/cmit/HP_BCU.html) from the original on 2015-01-12. Retrieved 2015-01-12.
10. See Intel® 64 and IA-32 Architectures Software Developer's Manual (http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html) Archived (https://web.archive.org/web/20120126002939/http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html) 2012-01-26 at the Wayback Machine, volume 3, section 9.1.2
11. page 5-27 *IBM Personal Computer Hardware Reference Library Technical Reference*, 1984, publication number 6361459
12. "IBM 5162 PC XT286 TechRef 68X2537 Technical Reference manual" (http://www.reenigne.org/crtc/PC-XT.pdf) (PDF). August 1986. p. 35 (System BIOS A-5). Archived (https://web.archive.org/web/20141211141709/http://www.reenigne.org/crtc/PC-XT.pdf) (PDF) from the original on 2014-12-11. Retrieved 2014-12-11.
13. How Stuff Works: What BIOS Does (http://computer.howstuffworks.com/bios1.htm) Archived (https://web.archive.org/web/20080207035123/http://computer.howstuffworks.com/bios1.htm) 2008-02-07 at the Wayback Machine.
14. *BIOS Boot Specification (Version 1.01, 11 January 1996)* (http://www.acpica.org/documentation/related_documents.php) Archived (https://web.archive.org/web/20120422184621/http://www.acpica.org/documentation/related_documents.php) April 22, 2012, at the Wayback Machine
15. Mueller, Scott (2001-06-08). "Processor Update Feature | Microprocessor Types and Specifications" (http://www.informit.com/articles/article.aspx?p=130978&seqNum=22) InformIT. Archived (https://web.archive.org/web/20140416183228/http://www.informit.com/articles/article.aspx?p=130978&seqNum=22) from the original on 2014-04-16. Retrieved 2014-04-15.
16. "Linux* Processor Microcode Data File" (https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=18148) *Download Center*. Downloadcenter.intel.com. 2009-09-23. Archived (https://web.archive.org/web/20140416180302/https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=18148) from the original on 2014-04-16. Retrieved 2014-04-15.
17. Scott Mueller, *Upgrading and repairing PCs 15th edition*, Que Publishing, 2003 ISBN 0-7897-2974-1, pages 109-110
18. "How SLP and SLIC Works" (http://www.guytechie.com/articles/2010/2/25/how-slp-and-slic-works.html) *guytechie.com*. 2010-02-25. Archived (https://web.archive.org/web/20150203135933/http://www.guytechie.com/articles/2010/2/25/how-slp-and-slic-works.html) from the original on 2015-02-03. Retrieved 2015-02-03.
19. "Create and add an OEM ACPI SLIC table module to a congatec BIOS" (http://www.congatec.com/fileadmin/user_upload/Documents/Application_Notes/AN21_Add_OEM_ACPI_SLIC_Table.pdf) (PDF). *congatec.com*. 2011-06-16. Archived (https://web.archive.org/web/20140802014113/http://www.congatec.com/fileadmin/user_upload/Documents/Application_Notes/AN21_Add_OEM_ACPI_SLIC_Table.pdf) (PDF) from the original on 2014-08-02. Retrieved 2015-02-03.
20. Whitson Gordon. "A Beginner's Introduction to Overclocking Your Intel Processor" (http://lifehacker.com/a-beginners-introduction-to-overclocking-your-intel-pr-5580998) *Lifehacker*. Gawker Media. Archived (https://web.archive.org/web/20141207213218/http://lifehacker.com/a-beginners-introduction-to-overclocking-your-intel-pr-5580998) from the original on 7 December 2014. Retrieved 6 December 2014.
21. Smart Computing Article - What Is The BIOS? (http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1994%2Fjuly94%2Fpcn0713%2Fpcn0713.asp) Archived (https://web.archive.org/web/20120310002756/http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1994%2Fjuly94%2Fpcn0713%2Fpcn0713.asp) 2012-03-10 at the Wayback Machine. - Computing Basics July 1994 • Vol.5 Issue 7
22. Torres, Gabriel (24 November 2004). "Introduction and Lithium Battery" (https://web.archive.org/web/20131224085334/http://www.hardwaresecrets.com/article/81). *Replacing the Motherboard Battery* hardwaresecrets.com. Archived from the original (http://www.hardwaresecrets.com/article/81) on 24 December 2013. Retrieved June 20, 2013.
23. "Decoding RAM & ROM (http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1997%2Fjun97%2F060997%2F060997.asp) Archived (https://web.archive.org/web/20120406173605/http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1997%2Fjun97%2F060997%2F060997.asp) 2012-04-06 at the Wayback Machine." *Smart Computing*. June 1997. Volume 8, Issue 6.

24. "Upgrading Your Flash BIOS For Plug And Play" (http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1996%2Fmar96%2F96n0324%2F96n0324.asp) Archived (https://web.archive.org/web/20120406173635/http://www.smartcomputing.com/editorial/article.asp?article=articles%2F1996%2Fmar96%2F96n0324%2F96n0324.asp) 2012-04-06 at the Wayback Machine." *Smart Computing*. March 1996. Volume 7, Issue 3.
25. "Time To Check BIOS (http://www.smartcomputing.com/editorial/article.asp?article=articles/archive/g0704/41u6/41u6.asp&guid=) Archived (https://web.archive.org/web/20110716092732/http://www.smartcomputing.com/editorial/article.asp?article=articles%2Farchive%2Fg0704%2F41u6%2F41u6.asp&guid=) 2011-07-16 at the Wayback Machine." *Smart Computing*. April 1999. Volume 7, Issue 4.
26. SplashTop's Instant-On Linux Desktop | Geek.com (http://www.geek.com/splashtops-instant-on-linux-desktop/) Archived (https://web.archive.org/web/20080907134251/http://www.geek.com/splashtops-instant-on-linux-desktop/) 2008-09-07 at the Wayback Machine.
27. 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*. Archived (https://web.archive.org/web/20120724081024/http://www.bit-tech.net/custompc/features/601716/the-life-and-times-of-the-modern-motherboard/page1.html) from the original on 24 July 2012 Retrieved 2 February 2013.
28. 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) (PDF). Intel. Archived (https://web.archive.org/web/20121018185412/http://download.intel.com/embedded/processor/whitepaper/322506.pdf) (PDF) from the original on 18 October 2012 Retrieved 2 February 2013.
29. Stiller, Andreas (2001). "Prozessor-Patches" (https://shop.heise.de/katalog/prozessor-patches) *c't* (in German). Heise (5): 240. Archived (https://web.archive.org/web/20151122084533/https://shop.heise.de/katalog/prozessor-patches) from the original on 2015-11-22 Retrieved 2015-11-21.
30. "Award BIOS logo" (http://fileformats.archiveteam.org/wiki/Award_BIOS_logo). 2015-06-15. Archived (https://web.archive.org/web/20151221152258/http://fileformats.archiveteam.org/wiki/Award_BIOS_logo) from the original on 2015-12-21. Retrieved 2015-12-06.
31. Phoenix Eagerly Waiting to Clone Next-Generation IBM BIOS (https://books.google.com/books?id=zzAEAAAAMBAJ&pg=PA8) Archived (https://web.archive.org/web/20140122080016/http://books.google.com/books?id=zzAEAAAAMBAJ&pg=PA8) 2014-01-22 at the Wayback Machine., *InfoWorld*, March 9, 1987
32. New BIOS Virus Withstands HDD Wipes (http://www.tomshardware.com/news/bios-virus-rootkit-security-backdoor,7400.html), March 27, 2009. Marcus Yam. Tom's Hardware US
33. Sacco, Anibal; Alfredo Ortéga. "Persistent BIOS Infection" (http://exploiting.wordpress.com/2009/03/23/cansecwest-was-great-here-the-presentation-slides/) *Exploiting Stuff*. Archived (https://web.archive.org/web/20090804105605/http://exploiting.wordpress.com/2009/03/23/cansecwest-was-great-here-the-presentation-slides/) from the original on 2009-08-04. Retrieved 2010-02-06.
34. Fisher, Dennis. "Researchers unveil persistent BIOS attack methods" (https://web.archive.org/web/20100130001722/http://threatpost.com/en_us/blogs/researchers-unveil-persistent-bios-attack-methods-031909) *Threat Post*. Archived from the original (http://threatpost.com/en_us/blogs/researchers-unveil-persistent-bios-attack-methods-031909) on 30 January 2010. Retrieved 2010-02-06.
35. 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*. Archived (https://web.archive.org/web/20110923143606/http://blog.webroot.com/2011/09/13/mebromi-the-first-bios-rootkit-in-the-wild/) from the original on 2011-09-23 Retrieved 2011-09-19.
36. "360发布"BMW病毒"技术分析报告" (https://web.archive.org/web/20110925051031/http://bbs.360.cn/4005462/251096134.html). *blog*. Archived from the original (http://bbs.360.cn/4005462/251096134.html) on 2011-09-25. Retrieved 2011-09-19.
37. Yuan, Liang. "Trojan.Mebromi" (http://www.symantec.com/security_response/writeup.jsp?docid=2011-090609-4557-99). *Threat Response*. Archived (https://web.archive.org/web/20110923145036/http://www.symantec.com/security_response/writeup.jsp?docid=2011-090609-4557-99) from the original on 2011-09-23 Retrieved 2011-09-19.
38. "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. Archived (https://web.archive.org/web/20140422042944/http://www.cbsnews.com/news/how-did-60-minutes-get-cameras-into-a-spy-agency/) from the original on 2014-04-22 Retrieved 2014-04-15.

39. Spencer Ackerman in Washington (2013-12-16). "NSA goes on 60 Minutes: the definitive facts behind CBS's flawed report | World news" (https://www.theguardian.com/world/2013/dec/16/nsa-surveillance-60-minutes-cbs-facts) theguardian.com. Archived (https://web.archive.org/web/20140125020846/http://www.theguardian.com/world/2013/dec/16/nsa-surveillance-60-minutes-cbs-facts) from the original on 2014-01-25. Retrieved 2014-01-27.
40. "Windows and GPT FAQ" (http://www.microsoft.com/whdc/device/storage/gpt_faq.mspx). *microsoft.com*. Microsoft. Archived (https://web.archive.org/web/20110219111649/http://www.microsoft.com/whdc/device/storage/GPT_FAQ.mspx) from the original on 19 February 2011. Retrieved 6 December 2014.
41. "Extensible Firmware Interface (EFI) and Unified EFI (UEFI)" (http://www.intel.com/technology/efi/) *Intel*. Archived (https://web.archive.org/web/20100105051711/http://www.intel.com/technology/efi/) from the original on 5 January 2010. Retrieved 6 December 2014.

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.
- Phoenix Technologies, Ltd. (June 1991). *System BIOS for IBM PCs, Compatibles, and EISA Computers — The Complete Guide to ROM-Based System Software*. Phoenix Technical Reference Series (2nd ed.). Amsterdam: Addison Wesley Publishing Company Inc. ISBN 0-201-57760-7.
- BIOS Disassembly Ninjutsu Uncovered, 1st edition, a freely available book in PDF format
- More Power To Firmware, free bonus chapter to the *Mac OS X Internals: A Systems Approach* book

External links

- List of BIOS options
- How BIOS Works
- Persistent BIOS Infection, *Phrack* #66, June 1, 2009, archived from the original on April 30, 2011
- Preventing BIOS Failures Using Intel Boot Block Flash Memory, Intel, December 1998, archived from the original on March 29, 2007
- BIOS Boot Specification 1.01, January 11, 1996
- Implementing a Plug and Play BIOS Using Intel's Boot Block Flash Memory, Intel, February 1995, archived from the original on November 28, 2007
