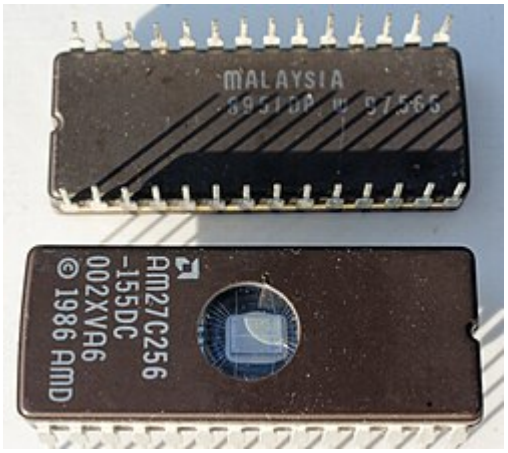
BIOS

By Contributors to Wikimedia projects

Published: 2001-10-27 · Archived: 2026-04-06 03:20:40 UTC

This article is about the BIOS as found in IBM PC/AT and compatibles. For the modern replacement that is often still called BIOS, see UEFI. For the general concept, see Firmware. For other uses, see Bios (disambiguation).



A pair of AMD BIOS chips for a Dell 310 computer from the 1980s. The bottom one shows the distinct window of an EPROM chip.

Year started	1981[a]
Organization	Originally IBM as proprietary software, later industry wide as a de facto standard. In 1996, the *BIOS Boot Specification* was written by Compaq, Phoenix Technologies and Intel.
Successor	UEFI

In computing, **BIOS** (, *BY-oss, -ohss*; **Basic Input/Output System**, also known as the **System BIOS**, **ROM BIOS**, **BIOS ROM** or **PC BIOS**) is a type of firmware used to provide runtime services for operating systems and programs and to perform hardware initialization during the booting process (power-on startup).[1] On a computer using BIOS firmware, the firmware comes pre-installed on the computer's motherboard.

The name originates from the **B**asic **I**nput/**O**utput **S**ystem used in the CP/M operating system in 1975.[2][3] The BIOS firmware was originally proprietary to the IBM PC; it was reverse engineered by some companies, such as Compaq, Phoenix Technologies, AMI and others, looking to create compatible systems. The interface of that original system serves as a *de facto* standard.

The BIOS in older PCs initializes and tests the system hardware components (power-on self-test or POST for short), and loads a boot loader from a mass storage device which then initializes a kernel. In the era of DOS, the BIOS provided BIOS interrupt calls for the keyboard, display, storage, 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 interrupt calls after startup.[4]

Most BIOS implementations are specifically designed to work with a particular computer or motherboard model, by interfacing with various devices especially system chipset. Originally, BIOS firmware was stored in a ROM chip on the PC motherboard. In later 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 could brick the motherboard.

Unified Extensible Firmware Interface (UEFI) is a successor to the PC BIOS, aiming to address its technical limitations.[5] UEFI firmware may include legacy BIOS compatibility to maintain compatibility with operating systems and option cards that do not support UEFI native operation.[6][7][8] Since 2020, all PCs for Intel platforms no longer support legacy BIOS.[9] The last version of Microsoft Windows to officially support running on PCs which use legacy BIOS firmware is Windows 10 as Windows 11 requires a UEFI-compliant system (except for IoT Enterprise editions of Windows 11 since version 24H2[10]).

```
/* 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 */
```

The term BIOS (Basic Input/Output System) was created by Gary Kildall[11][12] and first appeared in the CP/M operating system in 1975,[2][3][12][13][14][15] 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 the "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".

The BIOS originally proprietary to the IBM PC has been reverse engineered by some companies, such as Compaq, Phoenix Technologies, AMI and others, looking to create compatible systems.

With the introduction of PS/2 machines, IBM divided the System BIOS into real- 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.[16]

The BIOS of the original IBM PC and 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 expansion cards. Starting around the mid-1990s, it became typical for the BIOS ROM to include a *"BIOS configuration utility"* (BCU[17]) 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 floppy disk, not in the ROM, to set the configuration options contained in this memory. The floppy 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. The modern Wintel machine may store the BIOS configuration settings in flash ROM, perhaps the same flash ROM that holds the BIOS itself.

Extensions (option ROMs)

[edit]

Peripheral cards such as hard disk drive host bus adapters and video cards have their own firmware, and BIOS extension option ROM code may be a part of the expansion card firmware; that code provides additional capabilities in the BIOS. Code in option ROMs runs before the BIOS boots the operating system from mass storage. These ROMs typically test and initialize hardware, add new BIOS services, or replace existing BIOS services with their own services. For example, a SCSI controller usually has a BIOS extension ROM that adds support for hard drives connected through that controller. An extension ROM could in principle contain operating system, or it could implement an entirely different boot process such as network booting. 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 for initializing and bootstrapping integrated display and integrated storage. The initialization process can involve the execution of code related to the device being initialized, for locating the device, verifying the type of device, then establishing base registers, setting pointers, establishing interrupt vector tables,[18] selecting paging modes which are ways for organizing available registers in devices, setting default values for accessing software routines related to interrupts,[19] and setting the device's configuration using default values.[20] In addition, plug-in adapter cards such as SCSI, RAID, network interface cards, and video cards 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 a part of the motherboard BIOS.

An add-in card requires an option ROM if the card is not supported by the motherboard 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 boot process). 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.)

BIOS chips in a Dell 310 that were updated by replacing the chips

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.

The 8086 and 8088 start at physical address FFFF0h.[21] The 80286 starts at physical address FFFFF0h.[22] The 80386 and later x86 processors start at physical address FFFFFFF0h.[23][24][25] When the system is initialized, the first instruction of the BIOS appears at that address.

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 stored in nonvolatile BIOS memory ("CMOS") tested by the BIOS allows bypass of the lengthy POST and memory detection.

The POST identifies, tests and initializes system devices such as the CPU, chipset, RAM, motherboard, video card, keyboard, mouse, hard disk drive, optical disc drive and other hardware, including integrated peripherals.

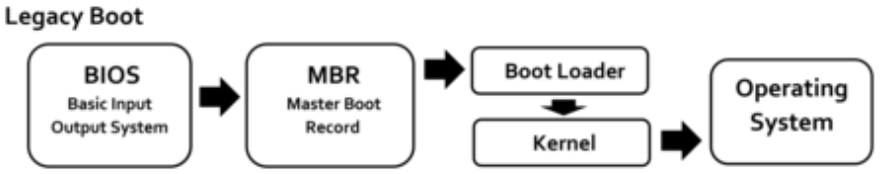
Early IBM PCs had a routine in the POST that would download a program into RAM through the keyboard port and run it.[26][27] This feature was intended for factory test or diagnostic purposes.

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 option ROMs, a BIOS implementation scans the real-mode address space from `0x0C0000` to `0x0F0000` on 2 KB (2,048 bytes) 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"). 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.

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. It is possible that an option ROM will not return to BIOS, pre-empting the BIOS's boot sequence altogether.



Boot process

After the POST completes and, in a BIOS that supports option ROMs, after the option ROM scan is completed and all detected ROM modules with valid checksums have been called, the BIOS calls interrupt 19h to start boot processing. Post-boot, programs loaded can also call interrupt 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 interrupt 19h is called, the BIOS attempts to locate boot loader software on 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.[28]

The BIOS uses the boot devices set in Nonvolatile BIOS memory (CMOS), or, in the earliest PCs, DIP switches. The BIOS checks each device in order to see if it is bootable by attempting to load the first sector (boot sector). If the sector cannot be read, the BIOS proceeds to 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.[b]

When a bootable device is found, the BIOS transfers control to the loaded sector. The BIOS does not interpret the contents of the boot sector other than to possibly check for the boot sector signature in the last two bytes. Interpretation of data structures like partition tables and BIOS Parameter Blocks is done by the boot program in the boot sector itself or by other programs loaded through the boot process.

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.

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. 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. The bootable virtual floppy disk can contain software that provides access to the optical medium in its native format.

If an expansion ROM wishes to change the way the system boots (such as from a network device or a SCSI adapter) 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.[*citation needed*]

Also, if an expansion ROM wishes to change the way the system boots unilaterally, it can simply hook interrupt 19h or other interrupts normally called from interrupt 19h, such as interrupt 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.[*citation needed*]

The user can select the boot priority implemented by the BIOS. For example, most computers have a hard disk that is bootable, but sometimes there is a removable-media drive that has higher boot priority, so the user can cause a removable disk to be booted.

In most modern BIOSes, the boot priority order can be configured by the user. 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 installed.

On the original IBM PC and XT, if no bootable disk was found, the BIOS would try to start ROM BASIC with the interrupt call to interrupt 18h. 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 interrupt 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 to retry the boot process. A modern BIOS may display nothing or may automatically enter the BIOS configuration utility when the boot process fails.

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 SS, SP, CS, and DL. CS:IP always points to physical address `0x07C00`. What values CS and IP actually have is not well defined. Some BIOSes use a CS:IP of `0x0000:0x7C00` while others may use `0x07C0:0x0000`.[29] Because boot programs are always loaded at this fixed address, there is no need for a boot program to be relocatable. DL may contain the drive number, as used with interrupt 13h, of the boot device. 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, 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, 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.[30][31]

Operating system services

[edit]

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 interrupt 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 were 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.

Modern operating systems, such as Windows and Linux, use the BIOS interrupt calls only during the booting process. 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.[4][*failed verification*]

Many 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 a CSM-capable UEFI firmware.

Processor microcode updates

[edit]

Intel processors have reprogrammable microcode since the P6 microarchitecture.[32][33][34] AMD processors have reprogrammable microcode since the K7 microarchitecture. The BIOS contains patches to the processor microcode that fix errors in the initial processor microcode; microcode is loaded into processor's SRAM so 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;[35] 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. Operating systems can also update the microcode.[36][37]

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 data table and contains no active code.[38][39]

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 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, 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.[38] 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.

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.[40]

Some older operating systems, for example MS-DOS, rely on the BIOS to carry out most input/output tasks within the PC.[41]

Calling real mode BIOS services directly is inefficient for protected mode (and long mode) operating systems. BIOS interrupt calls are not used by modern multitasking operating systems after they initially load.

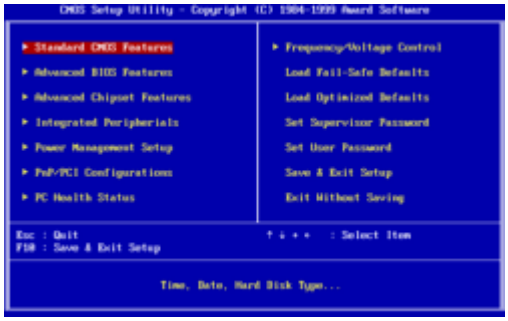
In the 1990s, BIOS provided some protected mode interfaces for Microsoft Windows and Unix-like operating systems, such as Advanced Power Management (APM), Plug and Play BIOS, Desktop Management Interface (DMI), VESA BIOS Extensions (VBE), e820 and MultiProcessor Specification (MPS). Starting from the year 2000, most BIOSes provide ACPI, SMBIOS, VBE and e820 interfaces for modern operating systems.[42][43][44][45][46]

After operating systems load, the System Management Mode code is still running in SMRAM. Since 2010, BIOS technology is in a transitional process toward UEFI.[5]

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 most IBM-compatibles with 80286 CPUs, had a battery-backed nonvolatile BIOS memory (CMOS RAM chip) that held BIOS settings.[47] 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.[48]

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 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.



Award BIOS setup utility on a standard PC

A modern BIOS setup utility has a text user interface (TUI) or graphical user interface (GUI) 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 DEL to enter Setup".

The actual key depends on specific hardware. The settings key is most often Delete (Acer, ASRock, Asus PC, ECS, Gigabyte, MSI, Zotac) and F2 (Asus motherboard, Dell, Lenovo laptop, Origin PC, Samsung, Toshiba), but it can also be F1 (Lenovo desktop) and F10 (HP).[49]

Features present in the BIOS setup utility typically include:

- Configuring, enabling and disabling the hardware components
- Setting the system time
- Setting the boot order
- Setting various passwords, such as a password for securing access to the BIOS user interface and preventing malicious users from booting the system from unauthorized portable storage devices, or a password for booting the system

Hardware monitoring

[edit]

A modern BIOS setup screen often features a **PC Health Status** or a **Hardware Monitoring** tab, which directly interfaces with a Hardware Monitor chip of the mainboard.[50] This makes it possible to monitor CPU and chassis temperature, the voltage provided by the power supply unit, as well as monitor and control the speed of the fans connected to the motherboard.

Once the system is booted, hardware monitoring and computer fan control is normally done directly by the Hardware Monitor chip itself, which can be a separate chip, interfaced through I²C or SMBus, or come as a part of a Super I/O solution, interfaced through Industry Standard Architecture (ISA) or Low Pin Count (LPC).[51] Some operating systems, like NetBSD with envsys and OpenBSD with sysctl hw.sensors, feature integrated interfacing with hardware monitors.

However, in some circumstances, the BIOS also provides the underlying information about hardware monitoring through ACPI, in which case, the operating system may be using ACPI to perform hardware monitoring.[52][53][54][55][56]

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

In modern PCs the BIOS is stored in rewritable EEPROM[57] or NOR flash memory,[58] allowing the contents to be replaced and modified. This rewriting of the contents is sometimes termed *flashing*. 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. Some computers also support updating the BIOS via an update floppy disk or a special partition on the hard drive.[59]

American Megatrends BIOS 686. This BIOS chip is housed in a PLCC package in a socket.

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,[60] but not altered, by users. To allow for updates, many compatible computers used re-programmable BIOS memory devices such as EPROM, EEPROM and later flash memory (usually NOR flash) 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.[61] BIOS versions are upgraded to take advantage of newer versions of hardware and to correct bugs in previous revisions of BIOSes.[62]

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, 31 December 1999.[63]

The first flash chips were attached to the ISA bus. Starting in 1998, the BIOS flash moved to the LPC bus, following a new standard implementation known as "firmware hub" (FWH). In 2005, the BIOS flash memory moved to the SPI bus.[64]

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 32 megabytes. For contrast, the original IBM PC BIOS was contained in an 8 KB 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 notebooks included Splashtop OS embedded into their NAND flash memory ICs.[65] 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.[*citation needed*]

The BIOS may contain components such as the Memory Reference Code (MRC), which is responsible for the memory initialization (e.g. SPD and memory timings initialization).[66]:8[67]

Modern BIOS[68] includes Intel Management Engine or AMD Platform Security Processor firmware.

Vendors and products

[edit]

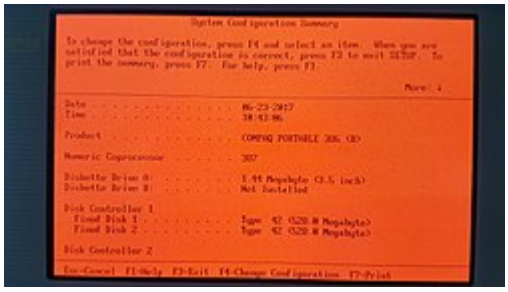
Comparison of different BIOS implementations

Company	AwardBIOS	AMIBIOS	Insyde	SeaBIOS
License	Proprietary	Proprietary	Proprietary	LGPL v3
Maintained / developed	Terminated	Terminated	Terminated	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[69])	Yes (LHA)	Yes (RLE)	Yes (LZMA)
CMOS	Yes	Yes	Yes	Yes
EDD	Yes	Yes	Yes	Yes
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
Splash screen	Yes (EPA)[70]	Yes (PCX)	Yes	Yes (BMP, JPG)
TPM	Unknown	Yes	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 full listings of the BIOSes 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. A benefit of the publication of the BIOS listings is that users could see exactly what the BIOS does and how it does it. However, those BIOSes were still copyright by IBM, so they could not be used by other manufacturers.

In March 1983, Compaq released its Compaq Portable, which included a clean-room engineered BIOS. This made Compaq became the first company that cloned an IBM PC successfully. However, Compaq did not offer their BIOS to other computer manufacturers.



Compaq Portable 386 BIOS

In May 1984, Phoenix Software Associates released its first ROM-BIOS. This BIOS enabled OEMs to build essentially fully compatible clones without having to reverse-engineer the IBM PC BIOS themselves, as Compaq had done for the Portable; it also helped fuel the growth in the PC-compatibles industry and sales of non-IBM versions of DOS.[71] The first American Megatrends (AMI) BIOS was released in 1986.

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.

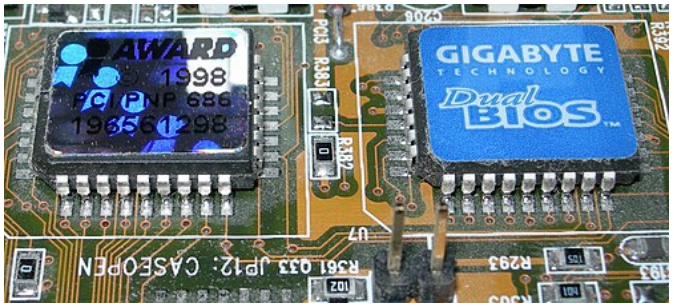
Many PC motherboard suppliers licensed the BIOS "core" and toolkit from a commercial third party, known as an "independent BIOS vendor" or IBV. The motherboard manufacturer then customized this BIOS to suit its own hardware. For this reason, updated BIOSes are normally obtained directly from the motherboard manufacturer. Major IBVs included American Megatrends (AMI), Insyde Software, Phoenix Technologies, and Byosoft. Microid Research and Award Software were acquired by Phoenix Technologies in 1998; Phoenix later phased out the Award brand name (although Award Software is still credited in AwardBIOS versions until 2001–2002, and AWRDACPI is still credited in Phoenix UEFI firmwares until late 2012). General Software, which was also acquired by Phoenix in 2007, sold BIOS for embedded systems based on Intel processors.

SeaBIOS is an open-source BIOS implementation.

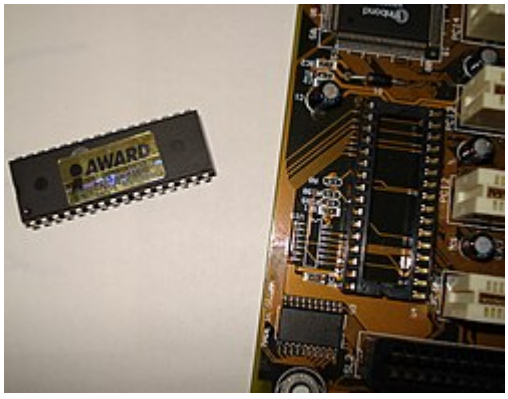
Open-source BIOS replacements

[edit]

The open-source community increased their effort to develop a replacement for proprietary BIOSes and their future incarnations with open-sourced counterparts. Open Firmware was an early attempt to make an open specification for boot firmware. It was initially endorsed by IEEE in its *IEEE 1275-1994* standard but was withdrawn in 2005.[72][73] Later examples include the OpenBIOS, coreboot and libreboot projects. AMD provided product specifications for some chipsets using coreboot, and Google is sponsoring the project. Motherboard manufacturer Tyan offers coreboot next to the standard BIOS with their Opteron line of motherboards.



Gigabyte DualBIOS PLCC32



A detached BIOS chip

EEPROM and flash memory 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 as DualBIOS) to recover from BIOS corruptions, and they use a special PLC to do BIOS recovery from BIOS corruptions.

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

The first BIOS virus was BIOS Meningitis, which instead of erasing BIOS chips it infected them. BIOS Meningitis was relatively harmless, compared to a virus like CIH.

The second BIOS virus was CIH, also known as the "Chernobyl Virus", which was able to erase flash ROM BIOS content on compatible chipsets. CIH appeared in mid-1998 and became active in April 1999. 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 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 using a hardware abstraction layer.[74]

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;[75] 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.[*citation needed*] 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.[*citation needed*]

The third 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.[76]

The fourth 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[77] 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."[78]

Mebromi is a trojan which targets computers with AwardBIOS, Microsoft Windows, and antivirus software from two Chinese companies: Rising Antivirus and Jiangmin KV Antivirus.[79][80][81] 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 the NSA had uncovered and thwarted a possible BIOS attack by a foreign nation state, targeting the US financial system.[82] The program cited anonymous sources alleging it was a Chinese plot.[82] However follow-up articles in *The Guardian*,[83] *The Atlantic*,[84] *Wired*[85] and *The Register*[86] refuted the NSA's claims.

Newer Intel platforms have Intel Boot Guard (IBG) technology enabled, this technology will check the BIOS digital signature at startup, and the IBG public key is fused into the PCH. End users can't disable this function.

Alternatives and successors

[edit]

For comparable software on other computer systems, see booting.

Unified Extensible Firmware Interface (UEFI) supplements the BIOS in many new machines. Initially written for the Intel Itanium architecture, UEFI is now available for x86 and Arm 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,[87] the Linux kernel 2.6.1 and later, and macOS on Intel-based Macs.[88] 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 legacy BIOS replacement in the open hardware community. UEFI is required for devices shipping with Windows 8[89][90] and above. Likewise, Windows 11 requires UEFI to boot,[91] with the exception of IoT Enterprise editions of Windows 11.[10]

After the popularity of UEFI in 2010s, the older BIOS that supported BIOS interrupt calls was renamed to "legacy BIOS".[*citation needed*]

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 the UEFI monitor in Intel-based Macs. As such, the BIOS utility no longer supports the legacy option, and prints "Legacy mode not supported on this system".

In 2017, Intel announced that it would remove legacy BIOS support by 2020. Since 2019, new Intel platform OEM PCs no longer support the legacy option.[92]

- Double boot
- Extended System Configuration Data (ESCD)
- Input/Output Control System
- ACPI (Advanced Configuration and Power Interface)
- 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)
- UEFI (Unified Extensible Firmware Interface)
- Das U-Boot, often used on embedded systems
- UBIOS

1. ^ Although the term BIOS predates 1981, the standard for IBM PC–compatible computers started with the release of the original IBM Personal Computer.
2. ^ 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 devices, 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.

1. ^ *Kozierok, Charles M. (2001-04-17). "Ref — System BIOS". The PC Guide. Archived from the original on 2019-02-18. Retrieved 2014-12-06.*
2. ^ Jump up to: *a b c Kildall, Gary Arlen (June 1975), CP/M 1.1 or 1.2 BIOS and BDOS for Lawrence Livermore Laboratories*
3. ^ Jump up to: *a b c Kildall, Gary Arlen (January 1980). "The History of CP/M - The Evolution of an Industry: One Person's Viewpoint" (Vol. 5, No. 1, Number 41 ed.). Dr. Dobb's Journal of Computer Calisthenics & Orthodontia. pp. 6–7. Archived from the original on 2016-11-24. Retrieved 2013-06-03.*
4. ^ Jump up to: *a b "Booting · Linux Inside". 0xax.gitbooks.io. Retrieved 2020-11-10.*
5. ^ Jump up to: *a b Bradley, Tony. "R.I.P. BIOS: A UEFI Primer". PCWorld. Archived from the original on 2014-01-27. Retrieved 2014-01-27.*
6. ^ *"Unified Extensible Firmware Interface". Intel.*
7. ^ *"UEFI". OSDev.org.*
8. ^ *"Intel® Platform Innovation Framework for EFI Compatibility Support Module Specification (revision 0.97)" (PDF). Intel. 2007-09-04. Retrieved 2013-10-06.*

9. ^ *"Removal of Legacy Boot Support for Intel Platforms Technical Advisory"*. *Retrieved 2024-07-25.*
10. ^ Jump up to: **a** **b** *"Minimum System Requirements for Windows IoT Enterprise"*. *Microsoft Learn*. *2024-05-22. Retrieved 2024-06-07.*
11. ^ *Swaine, Michael (1997-04-01). "Gary Kildall and Collegial Entrepreneurship". Dr. Dobb's Journal. Archived from the original on 2007-01-24. Retrieved 2006-11-20.*
12. ^ Jump up to: **a** **b** *"IEEE Milestone in Electrical Engineering and Computing - CP/M - Microcomputer Operating System, 1974" (PDF). Computer History Museum. 2014-04-25. Archived (PDF) from the original on 2019-04-03. Retrieved 2019-04-03.*
13. ^ *Shustek, Len (2016-08-02). "In His Own Words: Gary Kildall". Remarkable People. Computer History Museum. Archived from the original on 2016-12-17.*
14. ^ *Killian, A. Joseph "Joe" (2001). "Gary Kildall's CP/M: Some early CP/M history - 1976-1977". Thomas "Todd" Fischer, IMSAI. Archived from the original on 2012-12-29. Retrieved 2013-06-03.*
15. ^ *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" (PDF). Computer History Museum. Archived from the original (PDF) on 2014-07-14. Retrieved 2013-06-03.*
16. ^ *Glass, Brett (1989). "The IBM PC BIOS". Byte: 303–310. Retrieved 2021-12-31.*
17. ^ *"HP BIOS Configuration Utility". Hewlett-Packard. 2013. Archived from the original on 2015-01-12. Retrieved 2015-01-12.*
18. ^ *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.*
19. ^ *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.*
20. ^ *Sanchez, Julio; Canton, Maria P. (2003-02-26). The PC Graphics Handbook. CRC Press. ISBN 978-0-203-01053-2.*
21. ^ *"iAPX 86,88 User's Manual" (PDF). Intel. 1981. System Reset, p. 2-29, table 2-4. Retrieved 2025-08-05.*
22. ^ *"AMD 80286 Datasheet" (PDF). AMD. 1985. p. 13. "the 286 begins execution in real mode with the instruction at physical location FFFFF0H."*
23. ^ *"80386 Programmer's Reference Manual" (PDF). Intel. 1990. Section 10.1 Processor State After Reset, pages 10-1 - 10.3.*
24. ^ *"80386 Programmer's Reference Manual" (PDF). Intel. 1990. Section 10.2.3 First Instruction, p. 10-4. Retrieved 2013-11-03. "Execution begins with the instruction addressed by the initial contents of the CS and IP registers. To allow the initialization software to be placed in a ROM at the top of the address space, the high 12 bits of addresses issued for the code segment are set, until the first instruction which loads the CS register, such as a far jump or call. As a result, instruction fetching begins from address 0FFFFFFF0H."*
25. ^ *"Intel® 64 and IA-32 Architectures Software Developer's Manual" (PDF). Intel. May 2012. Section 9.1.4 First Instruction Executed, p. 2611. Archived from the original (PDF) on 2012-08-08. Retrieved 2012-08-23. "The first instruction that is fetched and executed following a hardware reset is located at physical*

address FFFFFFF0h. This address is 16 bytes below the processor's uppermost physical address. The EPROM containing the software-initialization code must be located at this address."

26. ^ page 5-27 *IBM Personal Computer Hardware Reference Library Technical Reference*, 1984, publication number 6361459
27. ^ *"IBM 5162 PC XT286 TechRef 68X2537 Technical Reference manual" (PDF). August 1986. p. 35 (System BIOS A-5). Archived (PDF) from the original on 2014-12-11. Retrieved 2014-12-11.*
28. ^ *"How StuffWorks: What BIOS Does". Archived from the original on 2008-02-07.*
29. ^ *Akeljic, Bekir (2017-01-01). "BIOS BASIC INPUT/ OUTPUT SYSTEM BIOS FUNCTIONS AND MODIFICATIONS". BIOS: 12. Archived from the original on 2022-08-08. Retrieved 2022-08-08 – via INTERNATIONAL UNIVERSITY TRAVNIKFACULITY OF INFORMATION TECHNOLOGY TRAVNIKSOFTWARE PROGRAMMING.*
30. ^ *"Memory Layout and Memory Map". flint.cs.yale.edu. Retrieved 2022-08-08.*
31. ^ *"BIOS Data ACPI Table (BDAT)" (PDF). Interface Specification. **4** (5): 67. 2020. Archived (PDF) from the original on 2021-07-03. Retrieved 2022-08-08.*
32. ^ *Stiller, Andreas; Paul, Matthias R. (1996-05-12). "Prozessorgeflüster". c't – magazin für computertechnik. Trends & News / aktuell - Prozessoren (in German). Vol. 1996, no. 6. Verlag Heinz Heise GmbH & Co KG. p. 20. ISSN 0724-8679. Archived from the original on 2017-08-28. Retrieved 2017-08-28.*
33. ^ *Mueller, Scott (2001-06-08). Processor Update Feature | Microprocessor Types and Specifications. InformIT. Archived from the original on 2014-04-16. Retrieved 2014-04-15.*
34. ^ *"Linux* Processor Microcode Data File". Download Center. Downloadcenter.intel.com. 2009-09-23. Archived from the original on 2014-04-16. Retrieved 2014-04-15.*
35. ^ *Scott Mueller (2003). Upgrading and repairing PCs 15th edition. Que Publishing. pp. 109–110. ISBN 0-7897-2974-1.*
36. ^ *"KB4100347: Intel microcode updates". support.microsoft.com. Retrieved 2020-09-20.*
37. ^ *"Microcode - Debian Wiki". wiki.debian.org. Retrieved 2020-09-19.*
38. ^ Jump up to: ***a*** ***b*** *"How SLP and SLIC Works". guytechie.com. 2010-02-25. Archived from the original on 2015-02-03. Retrieved 2015-02-03.*
39. ^ *"Create and add an OEM ACPI SLIC table module to a congatec BIOS" (PDF). congatec.com. 2011-06-16. Archived (PDF) from the original on 2014-08-02. Retrieved 2015-02-03.*
40. ^ *Whitson Gordon (2014-01-13). "A Beginner's Introduction to Overclocking Your Intel Processor". Lifehacker. Gawker Media. Archived from the original on 2014-12-07. Retrieved 2014-12-06.*
41. ^ *"Smart Computing Article - What Is The BIOS?". Computing Basics. Vol. 5, no. 7. July 1994. Archived from the original on 2012-03-10.*
42. ^ *"What is ACPI (Advanced Configuration and Power Interface)? - Definition from WhatIs.com". SearchWindowsServer. Retrieved 2020-09-18.*
43. ^ *"Changing hardware abstraction layer in Windows 2000 / XP – Smallvoid.com". 2001-01-15. Retrieved 2020-09-18.*
44. ^ *"What is ACPI?". www.spo-comm.de. Archived from the original on 2021-03-05. Retrieved 2020-09-18.*
45. ^ *lorihollasch. "Support for headless systems - Windows drivers". docs.microsoft.com. Retrieved 2020-12-05.*
46. ^ *"Memory Map (x86) - OSDev Wiki". wiki.osdev.org. Retrieved 2020-12-11.*

47. ^ *Torres, Gabriel (2004-11-24). "Introduction and Lithium Battery". Replacing the Motherboard Battery. hardwaresecrets.com. Archived from the original on 2013-12-24. Retrieved 2013-06-20.*
48. ^ *Beales, R. P. (2006-08-11). PC Systems, Installation and Maintenance. Routledge. ISBN 978-1-136-37441-8.*
49. ^ *"How to Enter the BIOS on Any PC: Access Keys by Manufacturer". Tom's Hardware. 2022-02-04.*
50. ^ *Constantine A. Murenin (2010-05-21). "11.1. Interfacing from the BIOS". OpenBSD Hardware Sensors – Environmental Monitoring and Fan Control (MMath thesis). University of Waterloo: UWSpace. hdl:10012/5234. Document ID: ab71498b6b1a60ff817b29d56997a418.*
51. ^ *Constantine A. Murenin (2007-04-17). "2. Hardware review". Generalised Interfacing with Microprocessor System Hardware Monitors. Proceedings of 2007 IEEE International Conference on Networking, Sensing and Control, 15–17 April 2007. London, United Kingdom: IEEE. pp. 901–906. doi:10.1109/ICNSC.2007.372901. ISBN 978-1-4244-1076-7. IEEE ICNSC 2007, pp. 901—906.*
52. ^ `aibs(4)` – OpenBSD Kernel Interfaces Manual
53. ^ `aibs(4)` – DragonFly BSD Devices and Device Drivers Manual
54. ^ `aibs(4)` – NetBSD Kernel Interfaces Manual
55. ^ `aibs(4)` – FreeBSD Kernel Interfaces Manual
56. ^ `acpi_thermal(4)` – FreeBSD Kernel Interfaces Manual
57. ^ *Clarke, Glen E.; Tetz, Edward (2007-01-30). CompTIA A+ Certification All-In-One Desk Reference for Dummies. John Wiley & Sons. ISBN 978-0-471-74811-3.*
58. ^ *Micheloni, Rino; Crippa, Luca; Marelli, Alessia (2010-07-27). Inside NAND Flash Memories. Springer. ISBN 978-90-481-9431-5.*
59. ^ *Mueller, Scott (2004). Upgrading and Repairing PCS. Que. ISBN 978-0-7897-2974-3.*
60. ^ *Clarke, Glen E.; Tetz, Edward (2007-01-30). CompTIA A+ Certification All-In-One Desk Reference for Dummies. John Wiley & Sons. ISBN 978-0-471-74811-3.*
61. ^ *"Decoding RAM & ROM". Smart Computing. Vol. 8, no. 6. June 1997. Archived from the original on 2012-04-06.*
62. ^ *"Upgrading Your Flash BIOS For Plug And Play". Smart Computing. March 1996. Archived from the original on 2012-04-06.*
63. ^ *"Time To Check BIOS". Smart Computing. April 1999. Archived from the original on 2011-07-16.*
64. ^ *"BIOS Flash Location". Archived from the original on 2021-08-18. Retrieved 2025-03-30.*
65. ^ *"SplashTop's Instant-On Linux Desktop | Geek.com". Archived from the original on 2008-09-07.*
66. ^ *Alex Watson (2007-11-27). "The life and times of the modern motherboard". Archived from the original on 2007-12-29.*
67. ^ *David Hilber Jr. (August 2009). "Considerations for Designing an Embedded Intel Architecture System with System Memory Down" (PDF). Intel. Archived (PDF) from the original on 2012-10-18. Retrieved 2013-02-02.*
68. ^ *"Types of BIOS". rompacks.com. Retrieved 2021-09-20.*
69. ^ *Stiller, Andreas (2001). "Prozessor-Patches". c't (in German) (5). Heise: 240. Archived from the original on 2015-11-22. Retrieved 2015-11-21.*
70. ^ *"Award BIOS logo". 2015-06-15. Archived from the original on 2015-12-21. Retrieved 2015-12-06.*
71. ^ *"Phoenix Eagerly Waiting to Clone Next-Generation IBM BIOS". InfoWorld. Vol. 9, no. 10. 1987-03-09. p. 8. Archived from the original on 2014-01-22.*

72. ^ *IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices. 1994-10-28. pp. 1–262. doi:10.1109/IEEESTD.1994.89427. ISBN 978-0-7381-1194-0. IEEE STD 1275-1994.*
73. ^ *"IEEE Standards Association". IEEE Standards Association. Retrieved 2023-04-26.*
74. ^ *"Definition of hardware abstraction layer". PCMAG. Retrieved 2022-07-11.*
75. ^ *Marcus Yam (2009-03-27). "New BIOS Virus Withstands HDD Wipes". Tom's Hardware US.*
76. ^ *"Black Hat 2006 Multimedia - Presentation, Audio and Video Archives". www.blackhat.com. Retrieved 2019-04-21.*
77. ^ *Sacco, Anibal; Alfredo Ortéga (2009-03-23). "Persistent BIOS Infection". Exploiting Stuff. Archived from the original on 2009-08-04. Retrieved 2010-02-06.*
78. ^ *Fisher, Dennis. "Researchers unveil persistent BIOS attack methods". Threat Post. Archived from the original on 2010-01-30. Retrieved 2010-02-06.*
79. ^ *Giuliani, Marco (2011-09-13). "Mebromi: the first BIOS rootkit in the wild". blog. Archived from the original on 2011-09-23. Retrieved 2011-09-19.*
80. ^ *"360发布"BMW病毒"技术分析报告". blog. Archived from the original on 2011-09-25. Retrieved 2011-09-19.*
81. ^ *Yuan, Liang. "Trojan.Mebromi". Threat Response. Archived from the original on 2011-09-23. Retrieved 2011-09-19.*
82. ^ Jump up to: ***a*** ***b*** *"How did 60 Minutes get cameras into a spy agency?". CBS News. 2013-12-15. Archived from the original on 2014-04-22. Retrieved 2014-04-15.*
83. ^ *Spencer Ackerman in Washington (2013-12-16). "NSA goes on 60 Minutes: the definitive facts behind CBS's flawed report". theguardian.com. Archived from the original on 2014-01-25. Retrieved 2014-01-27.*
84. ^ *Friedersdorf, Conor (2013-12-16). "A Question for 60 Minutes: Why Would China Want to Destroy the Global Economy?". The Atlantic. Retrieved 2019-03-26.*
85. ^ *Poulsen, Kevin (2013-12-16). "60 Minutes Puff Piece Claims NSA Saved U.S. From Cyberterrorism". Wired. ISSN 1059-1028. Retrieved 2019-03-26 – via www.wired.com.*
86. ^ *Sharwood, Simon (2013-12-16). "NSA alleges 'BIOS plot to destroy PCs'". The Register. Retrieved 2019-03-26.*
87. ^ *"Windows and GPT FAQ". microsoft.com. Microsoft. Archived from the original on 2011-02-19. Retrieved 2014-12-06.*
88. ^ *"Extensible Firmware Interface (EFI) and Unified EFI (UEFI)". Intel. Archived from the original on 2010-01-05. Retrieved 2014-12-06.*
89. ^ *"Next-gen boot spec could forever lock Linux off Windows 8 PCS".*
90. ^ *"Windows 8 secure boot could complicate Linux installs". 2011-09-21.*
91. ^ *"Windows 11 Specs and System Requirements | Microsoft". Microsoft. Retrieved 2021-10-14.*
92. ^ *Tung, Liam (2017-11-20). "Intel: We're ending all legacy BIOS support by 2020". ZDNET.*

- *IBM Personal Computer Technical Reference (Revised ed.). IBM Corporation. March 1983.*
- *IBM Personal Computer AT Technical Reference. IBM Personal Computer Hardware Reference Library. Vol. 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 [1]
- More Power To Firmware, free bonus chapter to the *Mac OS X Internals: A Systems Approach* book

Wikimedia Commons has media related to the BIOS.

Look up ***BIOS*** in Wiktionary, the free dictionary.

- *"BIOS Boot Specification 1.01" (PDF). Phoenix.com. 1996-01-11. Archived from the original (PDF) on 2011-07-15.*
- *"How BIOS Works". How Stuff Works. 2000-09-06.*
- *"Implementing a Plug and Play BIOS Using Intel's Boot Block Flash Memory" (PDF). Intel. February 1995. Archived from the original (PDF) on 2007-11-28. Retrieved 2007-11-28.*
- *"List of BIOS options". techarp.com. Archived from the original on 2014-01-27.*
- *"Persistent BIOS Infection". Phrack. No. 66. 2009-06-01. Archived from the original on 2011-04-30. Retrieved 2011-04-30.*
- *"Preventing BIOS Failures Using Intel Boot Block Flash Memory" (PDF). Intel. December 1998. Archived from the original (PDF) on 2007-03-29. Retrieved 2007-03-29.*
