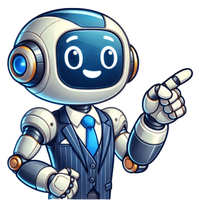
Click to verify



Firmware for hardware initialization and OS runtime servicesThis 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 started1981[a]OrganizationOriginally 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.SuccessorUEFIIn computing, BIOS (/baɪ.ɒs/, 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] The firmware comes pre-installed on the computer's motherboard.The name originates from the Basic Input/Output System 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 Phoenix Technologies) 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 */[I]* 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 1.2 for Lawrence Livermore Laboratories (LLL)[2]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 Phoenix Technologies) looking to create compatible systems.With the introduction of PS/2 machines, IBM offered two BIOS options, the "Personal Computer BIOS" for real-mode operating systems such as DOS, and the "Operating System BIOS" for protected-mode operating systems such as OS/2.[16]The BIOS of the original IBM PC and XT had no 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 PCsincluding the IBM ATheld 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.See also: Option ROMPeripheral 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. Code in option ROMs typically test and initialize hardware, add options to the BIOS interrupt calls, or 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 chipsOption 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, 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 FFFFF0h.[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 BIOS extension ROM it finds. A BIOS implementation scans the real-mode address space from 0x0C0000 to 0x0F0000 on 2KB (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 processAfter 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 successfully, some BIOSes will also check for the boot sector signature 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 other interrupts (for example, interrupt 18h for boot devices other than hard disks, or interrupt 19h for other bootstrap devices) and then force a reboot when it is ready for the system to be booted.After booting, programs can still interact with the BIOS to take advantage of its ability to abstract the hardware into a standard set of interfaces. Such programs can call BIOS interrupts to get information about the system configuration or to read or write blocks to the boot devices. Once booted, most operating systems do not call the BIOS unless booting a secondary program is required (such as a boot loader for another operating system). Instead, they use their own device drivers to access the hardware, but they may still use the BIOS to get information about the system configuration that they do not have built-in knowledge of. Some programs, such as memory testers, may also call BIOS interrupts to perform hardware diagnostics or to access hardware that they do not have built-in support for.Interrupts and timers[29]Interrupt 19h or other interrupts normally called from interrupt 19h, such as interrupt 13h, the BIOS disk service, to interrupt 18h, the BIOS ROM BASIC, or interrupt 1Ch, the BIOS timer service, can be used to gain control during the BIOS boot process. Then it can reset the BIOS boot process. Then it can reset the BIOS boot process, or it can merely trigger an alternate boot process such as booting from a different device than the one specified in the BIOS configuration. Interrupts 18h and 19h are also used by the BIOS ROM BASIC to indicate that the ROM BASIC should be loaded into RAM and run, if ROM BASIC is installed and if the BIOS boot process fails to find a bootable device. Interrupt 1Ch is called by the BIOS timer service to allow other programs to update themselves when the system time changes.Keyboard[30]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 0x00400to0x04FF 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.[31][32]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.In modern PCs running modern operating systems (such as Windows and Linux) the BIOS interrupt calls are used only during booting and initial loading of operating systems. 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]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.Intel processors have reprogrammable microcode since the P6 microarchitecture.[33][34][35] AMD processors have reprogrammable microcode since the K7 microarchitecture. The BIOS contain 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;[36] 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 update main processor microcode also.[37][38]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.[39][40]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.[39] 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.[41]Some older operating systems, for example MS-DOS, rely on the BIOS to carry out most input/output tasks within the PC.[42]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.[43][44][45][46][47]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.[48] 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.[49]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 PCA 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 (Ausmotherboard, Dell, Lenovo laptop, Origin PC, Samsung, Toshiba), but it can also be F1 (Lenovodesktop) and F10 (HP).[50]Features present in the BIOS setup utility typically include:Configuring, enabling and disabling the hardware componentsSetting the system timeSetting the boot orderSetting 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 systemMain article: Hardware monitoringA 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 motherboard.[51] 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 through the appropriate chip, interfaced through I²C or SMBus, or come as a part of a Superior I/O solution.Interfaces[52] Some operating systems, like Microsoft 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.[53][54]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[55] or NOR flash memory,[56] 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.[57]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,[58] 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.[59] BIOS versions are upgraded to take advantage of newer versions of hardware and to correct bugs in previous revisions of BIOSes.[60]Beginning with the IBM AT, PCs supported a hardware clock settable through BIOS; and a century byte 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.[61]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.[62]The size of the BIOS ROM has grown over time from 2 KB on the original IBM PC to 16 MB on some modern motherboards. Intel's Extensible Firmware Interface specification calls for BIOS extensions to be 1 MB or smaller in size, and many BIOS extensions fit this specification. Intel has also released a reference implementation BIOS with a size of 128 KB, and some BIOS extensions derive from Intel reference code or other open-source projects such as the Linux kernel. Some BIOS extensions are 32 KB in size because they contain 32-bit code, but they still use the BIOS APIs to access the computer hardware in real mode, and they may be limited by the BIOS APIs that they call even though they may switch to protected mode internally; such extensions may also contain 16-bit code for compatibility with the BIOS APIs they use, and they may also store some data in the BIOS ROM area in the memory space that is not used by the BIOS.Some BIOS extensions may be able to be stored in other types of ROM chips in the computer, such as in their NAND flash memory ICs.[63] However, the effect of including an operating system and/or an application program in a BIOS extension may be limited to the BIOS boot process and to providing services to the BIOS APIs during that process.[64]Nowadays, the BIOS is normally the only firmware in the PC that runs without the protection of a hardware-based memory protection system, but this is not necessarily the case in the future. Intel has released a specification that will eventually add memory protection to the BIOS and other firmware, and AMD has added optional memory protection to its AMD64 architecture BIOS firmware.Since the mid-1990s, 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. The AT, the keyboard interface is an embedded processor 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).[64]:8[65]Modern BIOS[66] includesIntel Management Engine or AMD Platform Security Processor firmware Comparison of different BIOS implementationsCompanyAwardBIOSAmiBIOSInsydeSeaBIOSIntelproprietaryProprietaryProprietaryLGPLsyMantecaderivedderivedTerminatedTerminatedTerminated[citation needed]Yes2-bit PCI BIOS calls[citation needed]YesYesYesYesYesYesYesYes(1.2)BBSYesYesYesYesYesYes