BASIC INPUT OUTPUT SYSTEM

[BIOS]

Seminar Presented by

Milind Chile - 2591
Dipti Borkar - 2778
Freddy Gandhi - 2787
Raghav Shreyas Murthi - 2804

Introduction

The BIOS, short for BASIC INPUT OUTPUT SYSTEM is a set of built-in software routines that give a PC its personality. Although, less than 32 kilobytes of code, the BIOS controls many of the most important functions of the PC: how it interprets keystrokes (Ctrl + Alt + Delete), how it puts characters on the screen, and how and at what speed it communicates through its ports. The BIOS also determines the compatibility of the computer and its flexibility in use. Although all BIOSs have the same function; all are not the same.

The BIOS governs the inner complexities arising out of the odd mixing of hardware and software. It acts as a link between the material hardware of the PC and its circuits, and the transcendent realm of software ideas and instructions. More than a link, the BIOS is both hardware and software. Like software, the BIOS is a set of instructions to the computer's microprocessor. Like hardware, however, these instructions are not evanescent; rather they are coded into the hard, worldly silicon of PROM, EPROM chips. Due to the twilight state of programs like the BIOS, existing in the netherworld between hardware and software, such PROM-based programs are often termed *firmware.*

The personality comes from the firmware code. This code determines how the computer will carry out the basic functions needed to make a working computer- how quickly they are carried out and how smoothly. In many PCs, this firmware also governs how the system board components interact, the chipset features that are used, even the amount of the microprocessor's time devoted to keeping memory working. The setup procedures in most new PCs are also held in the BIOS.

Every time the PC is switched on, the BIOS immediately takes command (control). The first thing it does is run through a series of diagnostic routines, system checks to ensure that every part of the PC is functioning correctly before any time or data is trusted to it. One by one, the BIOS checks the circuits of the system board and the memory, the keyboard, the disks and each expansion board. If a problem is found, it is reported with a code number on the monitor or as a coded series of beeps if an insufficient portion of the PC is functional to display anything on the monitor.

After the PC is operational, the BIOS does not rest. Its firmware includes several sets of routines that programs call to carry out everyday functions- typing characters on the screen, reading keystrokes, timing events. Programmers can create grand designs without worrying about tiny details because; the basic library is there.

The BIOS contains a small version of BASIC called, Cassette Basic, the language if only minimally useful, is used to prompt the user with a message like "Non-System Disk or Disk Error" when booting without a system disk. Most people never use cassette BASIC, but its there just in case. The distinct parts of the BIOS operate separately and distinctly although the code for each is contained inside the same silicon chip. The BIOS operates like a set of small *terminate and stay-resident programs* that are always in memory. In this case, they are always in memory because we cannot get them out.

BIOS Purpose

The design of any computer requires that many of the hardware elements of the machine be located at specific addresses within the range of input/output ports of the computer. Other computer components may have registers of their own that are used in their control. Because of the number of separate components inside any computer, the potential number of possible variations is limitless. Software that attempts to control any of this hardware must correctly reach out to these registers. As long as all computers are crafted exactly the same, with the same port used for exactly the same hardware with exactly the same registers, there should be no problem.

With the first PC itself however, IBM reserved the right to alter the hardware at will. They made no guarantee that any of the ports or registers would be the same in any later computers. This was where the BIOS came in. IBM had envisioned that programs would never have to directly address hardware. Instead they would call up a software routine in the BIOS that has the addressing part of the instruction permanently set in its code. If a different hardware arrangement is used then, the address inside the routines would be changed to match the updated hardware. The same software could thus work with a wide variety of hardware designs, giving the designer and the manufacturer the flexibility to upgrade the entirety of the system hardware should the need arise.

BIOS Shortcomings

The problem with BIOS routines is that no finite number of routines could possibly cover all situations and software needs optimally. Consequently, using BIOS routines is sometimes advantageous, but oftentimes a bother. In particular, BIOS routines can make many computer functions slow, and performance problems are most evident in the video display. For example, all IBM BIOS routines are designed for putting information on the video display one character at a time. Text can be blasted on the screen much faster by directly manipulating the hardware.

Using BIOS routines, software must first load particular registers with the character to be displayed along with its attribute (color, underline or the like...) and perhaps even its location on the screen. Then program issues a software interrupt to give the BIOS control to do its job. The BIOS then runs through a dozen or more assembly language instructions to move the character on the screen.

```
Mov ah, 00h
Mov al, 03h    ; sets 320 x 200 resolution
Int 10h
Mov edx, ycor  ; put y co-ordinate in EDX
Mov ecx, xcor  ; put y co-ordinate in ECX
Mov eax, 00000c00h
Add eax, color ; put color in EAX
Int 10h        ; software interrupt to call plot routine
```

Taking direct control – avoiding the BIOS – means writing directly to the display memory on the video card. A program can write directly to the screen just by loading the appropriate address and moving the needed byte value to that address in one assembly language step. The dozens of steps saved in writing each character add up to real performance gains, the difference between watching changes slowly scroll down the screen and instant updates.

Another limitation imposed by handling all system operations through the BIOS is that the computer cannot do anything without the BIOS knowing. For example, in standard modes the BIOS routines function well and allow reading, writing and formatting of disks in IBM formats. But it won't allow us to read or format a disk in any other format although; the disk drive has the capability to do so. Disk drives are more versatile than what BIOS makes us believe and making full use of the disk drive's capability means sidestepping the BIOS.

Direct Hardware Control

Bypassing the BIOS with programs that directly address the system hardware isn't difficult even when such a concept is forbidden by the IBM dream. In fact, so many software writers have taken their liberties with direct hardware control that many of hardware features of PCs are more standardized than the BIOS. Most prominent among these is the display memory. Serial ports too have developed beyond BIOS control. Every program that uses serial ports at speeds higher than 9600 bps (19200 bps on some machines) must sidestep the BIOS's serial communication routines.

Nevertheless, the BIOS offers other advantages to programmers. In many cases, using BIOS routines can simplify the writing of a program. Certain system operations are always available and can easily be accessed through software. These routines are usually well documented, well understood and bug-free, removing many of the concerns and worries of the programmer.

BIOS Compatibility

The goal of the compatible computer manufacture is to match the BIOS used by his machine with that inside the IBM AT. However, the code used by IBM is protected by copyright that forbids others from legally duplicating it. Instead compatible makers are charged with writing their own routine without copying IBM's. Few company companies have the resources to do it all themselves. Hence a vast majority of compatible PC manufacturers buy the necessary BIOS firmware from specialist firms such as AMI, Award Software, Phoenix Technologies and Mr. BIOS. And thus the cost of the manufacturer greatly decreases.

But since the exact code used by each BIOS version is different hence their compatibility with the IBM XT standard always varies. One of the biggest differences between these BIOS's is to do with " ENTRY POINTS " The various code routines in each BIOS start and end in addresses assigned to the BIOS function in the PC memory map. The address at which each routine starts is called the routines **Entry Point.** A few applications require that some entry points be at specific BIOS addresses. If the entry point of BIOS differs from what the program expects, the probable results are a system crash. Thus this compatibility issue must be dealt with.

Another compatibility issue with BIOSs is ensuring that a replacement BIOS is compatible with the computer in which you want to plug it. All BIOSs are created to match specific hardware. Uniting different hardware designs so that they can work interchangeably with all software will solve the problem.

Should you want to change or upgrade your PC's BIOS for any reason, you will need to get one that matches the exact of the computer you own.

BIOS performance

The BIOS in a PC can affect the system's performance in two ways.

a. The efficiency of the BIOS code.
b. The control it affords over the system resources.

a. Most of the programs do not know the contents of the BIOS routines and hence the assembly language instructions of each BIOS routine can vary considerably among different BIOSs. The most efficient BIOS routine will be the one with the least number of instructions. The program will hence require to execute fewer steps using fewer clock cycles every time it calls a BIOS routine. As a result the system runs faster. This is however only possible when a program takes advantage of BIOS routines and does not side step the BIOS to take direct hardware control.

b. A more important performance difference depends on upon how a BIOS initiates its host computer. Some BIOSs do a better job in optimizing the relationship between the Microprocessors local bus and the IO channel. A better BIOS automatically checks for the best operation of all available features. Through advanced setup procedures, it may also give the user manual control of these vital system parameters so that you can improve its settings.

Unfortunately, we have no way of knowing how well a BIOS works just by looking at a PC. The only way to judge is to run your application on the system and see what it does and how fast it does it.

BIOS Operation

The IBM BIOS is designed to work through a system of software interrupts. To activate a routine, a program issues the appropriate interrupt from the table below.

Interrupt in Hexadecimal	**Function**
00	Divide by zero
01	Single step
02	NMI
03	Break point
04	Over flow
05	Print screen
06	Reserved
07	Reserved
08	System Timer
09	Key Board
0A	Reserved
0B	Reserved
0C	Reserved
0D	Reserved
0E	Floppy Disk
0F	Reserved
10	Video
11	Equipment determination
12	Memory size determination
13	Floppy Disk
14	Asynchronous Communication
15	System services

16	Keyboard
17	Printer
19	Boot Strap Loader
1A	System Timer & real Time clock services
1B	Keyboard Break
1C	User Timer Tick
1D	Video Parameters
1E	Floppy Disk Parameters
1F	Video Graphic Characters
20 – 3F	Reserved for DOS
40	Floppy Disk BIOS revector
41	Hard Disk Parameters
42	Reserved
43	Reserved
44	Reserved
45	Reserved
46	Hard Disk Parameters
47	Reserved
48	Reserved
49	Reserved
4A	User Alarm
4B – 5F	Reserved
60 – 67	Reserved for user program interrupts
68 – 6F	Reserved
70	Real Time clock interrupt
71 – 74	Reserved
75	Redirect to NMI
76 – 7F	Reserved
80 – 85	Reserved for BASIC
86 – F0	Used by BASIC interpreter while Running BASIC
F1 – FF	Reserved for User program interrupts

The software interrupt causes the microprocessor to stop what it is doing and start a new routine, by saving the workspace. Each interrupt vector is a pointer that tells the microprocessor the location where the code associated with the interrupt is located.

The table of interrupt vectors begins at the very start of the microprocessors memory address 00000. Programs can order these vectors to change the meaning of the software interrupts.

Extendibility

The IBM BIOS gains much of its versatility by being an extendable BIOS. That is, the full extent of the BIOS is not cast forever in the silicon of the single PROM chip holding the firmware. The IBM BIOS can accept additional code as its own into one integrated whole. Hence additional PROM chips containing BIOS routines can be added to the PC. The BIOS will incorporate these new routines.

The key for making BIOS extendable is a Firmware routine that enables the BIOS to look for add-in code. During the boot up, BIOS code reads through the address range that is set aside for firmware looking for codes stored on add-in boards. If a valid section of code is found, the instructions are added to the BIOS repertory. For instance a new interrupt routine can be added or the functions of existing routines can be changed.

The routine of extending BIOS works as follows:

- **Search for Preamble Bytes**

  During POST after interrupt vectors have been loaded into RAM, the resident BIOS code instructs the computer to check its ROM memory for the occurrence of the special **preamble bytes**, that mark the beginning of add-in BIOS routines. The BIOS searches for these preamble bytes in the absolute address range 0C8000 – 0F4000.

- **Verification for legitimate BIOS extension**

  If the special preamble bits are found, it verifies that the subsequent section of code is a legitimate BIOS extension by performing a form of cyclic redundancy check on the specified number of 512 byte blocks. The values of each byte in the block are totaled using modulo 0100 addition – the effect is the same as dividing the sum of all the bytes by 4096 (d). A remainder of 0 indicates that the extension of BIOS contains valid code.

  The preamble bytes take a specific form.

  - ♣ Two bytes indicate the beginning of an extension code section: 055h followed by 0AAh.

  - ♣ Immediately following the two-byte preamble bytes is a third byte that gives the length of the additional BIOS. The number represents the amount of blocks 512 bytes long, needed to hold the extra code.

- **Installation of extension BIOS**

  After a valid section of code is identifies, system control (BIOS program execution) jumps to the 4th byte in the extension BIOS and performs any functions specified in the machine language. Typically these instructions tell the BIOS how to install the extra code.

- **Return of control to Resident BIOS**

  Finally when the instructions in the extension BIOS are completed, control returns to the resident BIOS. The system then continues to search for blocks of additional BIOS. When it completes its search by reaching the absolute address 0F4000 it starts the process of booting the computer from the disk.

The ROM chips containing this extra BIOS code need not be present on the system board. The memory locations used are also accessible on the extension bus. The code required to control the extension accessory thus loads when the system boots. One complication is that no two sections of code can occupy the same memory area. Consequently most expansion board makers for the PC series incorporate jumpers on their products to allow reassigning of the addresses used by their BIOS extensions to avoid conflicts.

Reading BIOS information

Nearly every BIOS includes some information about itself. This includes the copyright message so that you can determine the manufacturer and also the latest revision date so that one can identify how recently its code was updated.

The BIOS date is not just interesting but also a useful diagnostic tool. As PCs have expanded their capabilities, BIOSs were also revised to enable new operations. Sometimes solder versions of BIOSs do not work with new peripherals. Hence it is important to know the date of the BIOS in case the hardware is not working correctly. Most BIOS chips have their date and revision number printed on labels affixed over their EPROM windows. But we can also examine the BIOS date embedded in the BIOS code using the DEBUG program available in DOS.

Once you run the DEBUG program, you will get a hyphen prompt. Give the following command to read the contents of the memory location in BIOS.

D [address]:[offset] ; D command is for Dump/Display

The output is divided into three distinct parts horizontally:

a) Left part is the label of a memory location at which the display of 16 bytes begins
b) Central block gives the contents of each of the16 bytes of memory

c) Right block gives the ASCII representation of those values (if the value is a printable character). The date can be read in this right block as it is an ASCII representation.

(To exit DEBUG type command Q)

Example: We give the following command for the BIOS date.

D F000:FFF0

Output is:

F000:FFF0 CD 19 E0 00 F0 30 38 2F-32 35 2F 30 30 00 FC A5 **.....08/25/00...**

System Identification Bytes

All programs need to know the type of computer or system board on which they are attempting to run. Hence IBM had assigned one byte for this purpose. However now there are 2 bytes assigned to identify the system. These are known as ***Model Byte*** and the ***Submodel Byte.*** The model byte is located at absolute memory address 0FFFFE (hex) and the Submodel Byte follows it. Compatible computers usually use the value of system to which they are the closest match to a set of specific values of systems. Here is a list containing a few common systems.

SYSTEM	MODEL BYTE	SUBMODEL BYTE
PC	FF	-
XT	FE	-
Portable PC	FE	00
XT model 256	FC	02
PS/2 model 30	FA	00
PS/2 model 60	FC	04
Ps/2 model 80	F8	01

BIOS Data Area

After the BIOS code starts executing, it makes use of part of the host system's memory to store parameter values important to its operation.. Important among these include equipment flags, the base address of input/output adapters, keyboard characters and operating modes.

This BIOS data area comprises of 256 bytes of memory starting at absolute memory location 0000400 (hex). The following lists some of the interesting bytes in BIOS data area.

- Base address of RS232 adapters for COM1, COM2, COM3, COM4.
- Base address of printer adapters for LPT1, LPT2, LPT3.
- Number of floppy disk drives and number of hard disks installed
- Video modes and numeric coprocessor
- Keyboard status flags [location 0417]

BIT	KEY	Status
0	Right Shift	Pressed
1	Left Shift	Pressed
2	Control	Pressed
3	Alt	Pressed
4	Scroll Lock	Locked
5	Num Lock	Locked
6	Caps Lock	Locked
7	Insert Lock	Locked

- ♣ Pointer to head and tail of Keyboard buffer as well as the Keyboard buffer itself
- ♣ Last Disk Drive operation status flags [location 0441]

BIT	Status
0	No error
1	Invalid disk drive parameter
2	Address mark not found
3	Write protect error
4	Request sector not found
10	CRC error on disk read
20	General controller failure
40	Seek operation failure
80	Disk drive not ready

- ♣ Timeout counters for response of serial devices (COM1, COM2, COM3, COM4) and printers (LPT1, LPT2, LPT3).
- ♣ Calendar information (days count since jan1 1980)
- ♣ Rows and columns displayed on monitor

ROM BASIC:

One section of the BIOS code is usually not duplicated by computer makers since its is not only copyrighted but also undocumented as to function and entry points. This section is actually a primitive programming language called "***Cassette BASIC***" or sometimes called as ROM BASIC. The original purpose of the cassette BASIC language was to enable the first IBM computer to do anything without the need for a disk drive.

When you first boot your computer without a system disk, any software which is supposed to execute will have to load from some disk will fail and you usually get an error "non-system disk / error". But even to run this check program, IBM computers have to start the cassette BASIC language executing. All advanced versions of BASIC are designed to augment this cassette BASIC already in the computer's ROM.

CMOS

During the earlier days, all the differences between earlier PC's could be coded by one or two banks of DIP switches. As the options piled up, a number of switches couldn't meet the requirements. To overcome the above shortcoming, vital system parameters began to be stored in a special, small block of battery backed CMOS. The various system configurations (which include information about floppy disk, hard disk, the presence of co-possessor etc) are stored in the CMOS.

CMOS SETUP UTILITY

VIRUS WARNING:
With all the concern in the industry about computer viruses, some BIOS makers have added their own form of protection, warning message that appears when software attempts to write the Boot Sector of Hard Disk. When the protection is switched on, you're given the option of canceling the write operation to prevent infection. Some operating systems such as OS/2 Boot Manage rewrite boot sector data when you switch between boot modes, you can't simply obstruct all boot sector write operations.

NOTE: Disabling this while configuring or setting up a system, or it will drive you nuts interrupting you with warning messages when you partition and format your hard disk.

CACHE OPERATION
Some BIOSs allow you to switch on or off your Internal Cache inside your microprocessor and the external cache.

NOTE:
The only tome you should switch off (Disable) your system cache is when you want to pin down some software problems or diagnose some hardware errors.

QUICK POWER ON SELF-TEST

Enabling this setting will cause the BIOS power-on self test routine to skip some of its tests during bootup. One of the key things this setting usually does when enabled is cause the POST to skip checking all of extended memory for errors.

Most people enable this setting to speed up the boot process, but you should realize that you do increase the chance of the POST missing an error if you use this. Fortunately (or unfortunately) the POST memory test is virtually useless to detect transient memory errors (as opposed to hard errors that you would discover the first time you powered up the machine with the new memory in it), so once your system is running and stable, you can in most cases enable this setting safely. It' s still safest to leave it disabled, which is what I recommend unless you have truly monstrous amounts of RAM. After all, how often do you boot the system during normal use?

BOOT SEQUENCE

This option is to specify the boot order of your PC's disk drive, i.e the order in which the drives should be searched to find where the OS resides.

BOOT UP FLOPPY SEEK:

This BIOS option lets you toggle between yes or no. If one selects 'no", the BIOS will ignore the floppy disk drive when attempting to boot your PC, even if has a valid system disk.

NOTE: The advantage of this option is to prevent inexperience users from booting your PC from their own, possibly virus infected floppy disks.

SWAP FLOPPY DRIVES:

A useful feature for those machines that use two floppy drives, when enabled this swaps the A: and B: drives. This enables you to change the bootable floppy without having to open the case and switch the cable

MEMORY PARITY:

Some systems permit you to switch off memory parity checking, disabling error detection. Taking this option prevents your system from halting when memory parity errors are detected.

NOTE: If disabled and the error occurs within data, you may never know when your information is inaccurate.

TYPEMATIC RATE and TYPEMATIC DELAY:

With most keyboards, when you press down and hold a key, after a short delay the keyboard begins to send a continuous sequence of characters you've pressed, ending only when you release the key. This feature is called TYPEMATIC. You can control the speed at which the keyboard shoots out individual characters (Typematic Rate) and the delay before which Typematic Kicks in (Typematic Delay).

Typematic Rate is usually expressed in characters per second. Use what feels comfortable, but don' t go too high or you may feed the characters faster than the system can deal with them, which can cause beeping or even system lockups.

NOTE: Some higher-end keyboards have the ability to set this parameter built-in; sometimes it is called "Key Repeat" or "Repeat Rate".

ROM SHADOWING

In most PCs, there is a full 384 KB area of RAM in the UMA. When any addresses in the UMA region are used by ROMs, the RAM underlying them is hidden.

One problem with ROMs such as those used for the system BIOS and video BIOS, is that it is relatively slow. The access time of ROMs is usually between 120 and 200 ns, compared to system RAM which is typically 50 to 70 ns. Also, system RAM is accessed 32 bits at a time, while ROMs are usually 16 bits wide. The result of this is that accesses to the BIOS code are very slow relative to accesses to code in the system memory.4

I'm sure you can see where this is heading. Since there is RAM hiding underneath the ROMs anyway, most systems have the ability to "mirror" the ROM code into this RAM to improve performance. This is called *ROM Shadowing*, and is controlled using a set of BIOS parameters. There is normally a separate parameter to control the shadowing of the system BIOS, the video BIOS and adapter ROM areas.

NOTE:

When shadowing of a region of memory is enabled, at boot time the BIOS copies the contents of the ROM into the underlying RAM, write-protects the RAM and then disables the ROM. To the system the shadow RAM appears as if it is ROM, and it is also write-protected the way ROM is. This write-protection is important to remember, because if you enable shadowing of memory addresses that are being used for *RAM*, the device using it will cease to function when the RAM can no longer be written to (it is locked out by the shadowing). Some network cards for example use parts of the memory region they occupy for both ROM and RAM functions. Enabling shadowing there will cause the card to hang up due to the write-protection. Similarly, you should never turn on shadowing of the regions of memory being used for an EMS frame buffer or for UMBs.

In normal circumstances, the system BIOS and video BIOS are the only areas shadowed. This can in theory cause problems with some operating systems, though I have never personally encountered this. I have also heard rumors of video cards that don't function correctly when video BIOS shadowing is *off*, but I haven't encountered that myself either

VIDEO BIOS SHADOWING:

This parameter, when enabled, turns on BIOS ROM shadowing for the block of memory normally used for standard VGA video ROM code, which is C0000 to C7FFF (32K) in short, it speeds up your system by copying the contents of your video BIOS code from the slow ROM in which it resides into faster RAM.

The default for this setting depends on the particular system a great deal.. Enabling it will increase performance. Disable it if it causes system problems, particularly those related to the video subsystem.

NOTE: On some systems the video BIOS shadow setting is named for the address range the video BIOS occupies, C0000-C7FFFh, instead of being specifically called "Video BIOS Shadow".

SYSTEM BIOS SHADOWING

When enabled, this parameter turns on BIOS ROM shadowing for the block of memory that contains your system BIOS. This is normally F0000 to FFFFF (64K); in short, it speeds up your system by copying the contents of your system BIOS code from the slow ROM in which it resides into faster RAM.

C8000-CBFFF SHADOW, CC000-CFFFF SHADOW.

Expansion cards such as network adapters normally use the areas of memory from C8000 to DFFFFh. Turning on shadowing would speed these adapters up in the same way that shadowing the system BIOS speeds up the system BIOS code

PASSWORD

This section lets you set security passwords to control access to the system at boot time and/or when entering the BIOS setup program. Some systems have a single password, while many newer ones now have two: a supervisor and a user password.

Supervisor Password:

Select this option to set the supervisor password. The supervisor password is the higher-level password of the two normally present on the system. On most systems, when the supervisor password has been set, it must be entered in order to access the BIOS setup program, or to change the user password.

User Password

Select this option to set the user password. The user password is the lower level password of the two normally present on the system. The user password usually allows the system to be booted, but does not allow access to the BIOS setup program. The supervisor password must be used to enter the BIOS setup program.

NOTE: On some systems, either the supervisor or user passwords will allow access to the BIOS setup program. In this case the existence of two passwords may be to allow a single password to be set up for an administrator, which will work for multiple machines, while the user password is individual for each machine.

Trouble shooting / FAQ's on BIOS

1. What is a (Flash) BIOS ?

BIOS is an acronym for basic input/output system. The BIOS is built-in software that determines what a computer can do without accessing programs from a disk. On PCs, the BIOS contains all the code required to control the keyboard, display screen, disk drives, serial communications, and a number of miscellaneous functions.

The BIOS is typically placed on a ROM chip that comes with the computer (it is often called a ROM BIOS). This ensures that the BIOS will always be available and will not be damaged by disk failures. It also makes it possible for a computer to boot itself.

Because RAM is faster than ROM, many computer manufacturers design systems so that the BIOS is copied from ROM to RAM each time the computer is booted. This is known as shadowing, and should be disabled in the BIOS setup before flashing.

Most modern PCs have a flash BIOS, which means that the BIOS has been recorded on a rewriteable memory chip, which can be updated if necessary.

The PC BIOS is standardized; so all PCs are alike at this level (although there are different BIOS versions). Additional DOS functions are usually added through software modules. This means you can upgrade to a newer version of DOS without changing the BIOS.

PC BIOSes that can handle Plug-and-Play (PnP) devices are known as PnP BIOSes, or PnP-aware BIOSes. These BIOSes are always implemented with flash memory rather than ROM.

2. How do you flash your BIOS ?

To flash your BIOS you' ll need a) a flasher, and b) a data-file. The flasher programs the data-file into the BIOS chip.

Boot to the DOS prompt, either using a CLEAN boot disk or Safe Mode DOS Prompt. Type the following at the DOS prompt, where xxx is the name of the BIOS file you downloaded:

awdflash xxx.bin (for Award BIOSs)
amiflash xxx.bin (for AMI BIOSs)
mrflash xxx.bin (for MRBIOSs)

Notes: Most flashers will ask you to save the current BIOS. Choose Yes, so that you can always flash back to the original version if you' re having problems with the new one.

Some manufacturers may use their own utilities to upgrade the BIOS (mostly non-clones)
Disable the System BIOS Cacheable option in the BIOS before flashing.
Do NOT flash under Windows or any OS other than plain DOS.
By using the switch /? (eg. awdflash /?) the flasher will display all available switches.

3. Can something go wrong during flashing ?

Yes, if you use the wrong flash BIOS, or have a power outage, or have a defective chip, there is chance that your computer WILL NOT BOOT. We recommend not to flash unless absolutely necessary.

4. How can you recover a corrupt BIOS ?

Solution 1: Boot-block BIOS

Modern motherboards have a **boot-block BIOS**. This is small area of the BIOS that doesn't get overwritten when you flash a BIOS chip. The boot-block BIOS only has support for the floppy drive. If you have a PCI video card you won't see anything on the screen because the boot-block BIOS only supports an ISA video card.

Award: The boot-block BIOS will execute an AUTOEXEC.BAT file on a bootable diskette. Copy an Award flasher & the correct BIOS *.bin file on the floppy and execute it automatically by putting awdflash *.bin in the AUTOEXEC.BAT file.

AMI: The AMI boot-block BIOS will look for a AMIBOOT.ROM file on a diskette. Copy and rename the correct BIOS file on the floppy and power up the PC. The floppy doesn't need to be bootable. You will see the PC read the floppy, after about 4 minutes you will hear 4 beeps; this means the transfer is done. Reboot the PC and modify the CMOS for your configuration.

Solution 2: Get a new BIOS chip

Contact your motherboard manufacturer to see if they sell BIOS chips. Some motherboard manufacturers send them for free.
Contact a company that sells pre-flashed chips, like Unicore Software, FlashBIOS.ORG or BadFlash

Solution 3: Hot-swapping

Replace the corrupt chip by a working one. The best option is to take the working BIOS chip from a motherboard, which has the same chipset although that's not absolutely necessary. It just has to give you a chance of booting into DOS. Before pulling the working BIOS chip out of it's original motherboard, set the System BIOS cacheable option in the BIOS to enabled.

After you have put the working BIOS in the motherboard with the corrupt BIOS boot the system to DOS (with a floppy or HD).

Now replace (while the computer is powered on) the working BIOS chip with the corrupt one. Flash an appropriate BIOS to the corrupt BIOS and reboot.

Solution 3: (for Intel motherboards)

Change Flash Recovery jumper to the recovery mode position (not all products have this feature)

Install the bootable upgrade diskette into drive A:

Reboot the system

Because of the small amount of code available in the non-erasable boot block area, no video is available to direct the procedure. Listening to the speaker and looking at the floppy drive LED can monitor the procedure. When the system beeps and the floppy drive LED is lit, the system is copying the recovery code into the FLASH device. As soon as the drive LED goes off, the recovery is complete.
Turn the system off
Change the Flash Recovery jumper back to the default position
Leave the upgrade floppy in drive A: and turn the system on
Continue with the original upgrade

5. How can you clear all of the BIOS settings back to their defaults with the DOS DEBUG command ?

Solution 1: for AMI and Award BIOSs:

C:\DEBUG (at a clear DOS prompt, don't do this in DOS box in Windows)
-O 70 17
-O 71 17
-Q

Solution 2: for Phoenix BIOSs:

C:\DEBUG (at a clear DOS prompt, don't do this in DOS box in Windows)
-O 70 FF
-O 71 17
Q

6. What is the BIOS actually doing when I turn on my computer?

When you turn on your computer, several events occur automatically:

The CPU "wakes up" (has power) and reads the x86 code in the BIOS chip.
The code in the BIOS chip runs a series of tests, called the POST for Power On Self Test, to make sure the system devices are working correctly. In general, the BIOS:

Initializes system hardware and chipset registers
Initializes power management
Tests RAM (Random Access Memory)
Enables the keyboard
Tests serial and parallel ports
Initializes floppy disk drives and hard disk drive controllers
Displays system summary information

During POST, the BIOS compares the system configuration data obtained from POST with the system information stored on a CMOS - Complementary Metal-Oxide Semiconductor - memory chip located on the motherboard. (This CMOS chip, which is updated whenever new system components are added, contains the latest information about system components.)

After the POST tasks are completed, the BIOS looks for the boot program responsible for loading the operating system. Usually, the BIOS looks on the floppy disk drive A: followed by drive C:

After being loaded into memory, the boot program then loads the system configuration information (contained in the registry in a Windows® environment) and device drivers. Finally, the operating system is loaded, and, if this is a Windows® environment, the programs in the Start Up folder are executed.

Future of BIOS

Windows provides its own interface to computer hardware, and Windows programs are required to use the Windows interface. To allow the use of old, non-Windows programs that directly access PC hardware, Windows captures the commands meant for direct hardware control and routes them to its own emulation of the underlying hardware. These contrary programs interact only with the emulation, and Windows takes care of talking to the hardware. Technically speaking, Windows 3.x (and above) both virtualized and serialized direct hardware access by real-mode programs using Virtual-86 mode of the 80386 (and above) processors. This was particularly useful when multiple real-mode programs concurrently wanted direct access to a single hardware device.

Even though Windows takes care of these formerly vital BIOS functions, every PC still requires a BIOS. The basic BIOS interface functions are still required even in modern PCs because you need some control over your computer before the operating system loads. The keyboard must work so you can select options. The video system must come alive so you can monitor how what you do and what the system is doing. And the basic disk controls must work so your computer can load the operating system software from disk. Moreover, not everyone runs Windows. Programs that call upon BIOS services remain useful in some applications. Consequently, although little used by most people, the interface function of the BIOS survives and will likely long continue to do so