Boot Sector

The boot sector, located at sector 1 of each volume, is a critical disk structure for starting your computer. It contains executable code and data required by the code, including information that the file system uses to access the volume. The boot sector is created when you format a volume. At the end of the boot sector is a two-byte structure called a signature word or end of sector marker, which is always set to 0x55AA. On computers running Windows 2000, the boot sector on the active partition loads into memory and starts Ntldr, which loads the operating system.

The Windows 2000 boot sector consists of the following elements:

- An *x* 86-based CPU jump instruction.
- The original equipment manufacturer identification (OEM ID).
- The BIOS parameter block (BPB) a data structure.
- The extended BPB.
- The executable boot code (or bootstrap code) that starts the operating system.

Note

All Windows 2000 boot sectors contain these elements. However, the NTFS boot sector, the FAT16, and the FAT32 boot sectors are all formatted differently.

The BPB describes the physical parameters of the volume: the extended BPB begins immediately after the BPB. Due to differing types of fields and the amount of data they contain, the length of the BPB is different for FAT16, FAT32, and NTFS boot sectors.

The information in the BPB and the extended BPB is used by disk device drivers to read and configure volumes. The area following the extended BPB typically contains executable boot code, which performs the actions necessary to continue the startup process.

Boot Sector Startup Processes

Computers use the boot sector to run instructions during startup. The initial startup process is summarized in the following steps:

1. The system BIOS and the CPU initiate the power-on self test (POST).
2. The BIOS searches for a boot device (typically a disk).
3. The BIOS loads the first physical sector of the boot device into memory and transfers CPU execution to that memory address.

If the boot device is on a hard disk, the BIOS loads the MBR. The master boot code in the MBR loads the boot sector of the active partition, and transfers CPU execution to that memory address. On computers that are running Windows 2000, the executable boot code in the boot sector finds Ntldr, loads it into memory, and transfers execution to that file.

Note

Windows 2000 cannot start up from a spanned, striped, or RAID-5 volume that is running dynamic disk. These disk structures cannot be registered into the MBR's partition table, so a system partition using these structures is not startable. Windows 2000 must be fully loaded into memory before they can be used.

If there is a floppy disk in drive A, the system BIOS loads the first sector (the boot sector) of the disk into memory. If the disk is startable — formatted by MS-DOS with core operating system files applied — the boot sector loads into memory and uses the executable boot code to transfer CPU execution to Io.sys, a core MS-DOS operating system file. If the floppy disk is not bootable, the executable boot code displays an error message such as:

```
Non-System disk or disk error

Replace and press any key when ready
```

Note

This error will not appear on normally functioning systems that are configured to look for the startup files on drive C first. On many computers, an option in the CMOS setup program allows the user to set the sequence of installed disks that the system searches for the startup files.

If you get similar errors when trying to start the computer from the hard disk, the boot sector might be corrupted. For more information about troubleshooting boot sector problems, see "Damaged MBRs and Boot Sectors" later in this chapter.

Initially, the startup process is independent of disk format and operating system. The unique characteristics of operating and file systems become important when the boot sector's executable boot code starts.

Top Of Page

Components of a Boot Sector

The MBR transfers CPU execution to the boot sector, so the first three bytes of the boot sector must be valid, executable *x* 86-based CPU instructions. This includes a jump instruction that skips the next several nonexecutable bytes.

Following the jump instruction is the 8-byte OEM ID, a string of characters that identifies the name and version number of the operating system that formatted the volume. To preserve compatibility with MS-DOS, Windows 2000 records "MSDOS5.0" in this field on FAT16 and FAT32 disks. On NTFS disks, Windows 2000 records "NTFS."

Note

You may also see the OEM ID "MSWIN4.0" on disks formatted by Windows 95 and "MSWIN4.1" on disks formatted by Windows 95 OSR2 and Windows 98. Windows 2000 does not use the OEM ID field in the boot

sector except for verifying NTFS volumes.

Following the OEM ID is the BPB, which provides information that enables the executable boot code to locate Ntldr. The BPB always starts at the same offset, so standard parameters are in a known location. Disk size and geometry variables are encapsulated in the BPB. Because the first part of the boot sector is an x86 jump instruction, the BPB can be extended in the future by appending new information at the end. The jump instruction needs only a minor adjustment to accommodate this change. The BPB is stored in a packed (unaligned) format.

Top Of Page

FAT16 Boot Sector

Table 1.6 describes the boot sector of a volume formatted with the FAT16 file system.

Table 1.6 Boot Sector Sections on a FAT16 Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BPB
0x24	26 bytes	Extended BPB
0x3E	448 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker

The following example illustrates a hexadecimal printout of the boot sector on a FAT16 volume. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B– 0x3D are the BPB and the extended BPB.
- The remaining section is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 0, Side 1, Sector 1

00000000: EB 3C 90 4D 53 44 4F 53 - 35 2E 30 00 02 40 01 00 .<.MSDOS5.0 ..@..

00000010: 02 00 02 00 00 F8 FC 00 - 3F 00 40 00 3F 00 00 00 ........?.@.?...

00000020: 01 F0 3E 00 80 00 29 A8 - 8B 36 52 4E 4F 20 4E 41 ..>...)..6RNO NA

00000030: 4D 45 20 20 20 20 46 41 - 54 31 36 20 20 20 33 C0 ME FAT16 3.
```

```
00000040: 8E D0 BC 00 7C 68 C0 07 - 1F A0 10 00 F7 26 16 00 ....|h......&..
00000050: 03 06 0E 00 50 91 B8 20 - 00 F7 26 11 00 8B 1E 0B ....P.. ..&.....
00000060: 00 03 C3 48 F7 F3 03 C8 - 89 0E 08 02 68 00 10 07 ...H........h...
00000070: 33 DB 8F 06 13 02 89 1E - 15 02 0E E8 90 00 72 57 3.............rW
00000080: 33 DB 8B 0E 11 00 8B FB - 51 B9 0B 00 BE DC 01 F3 3.......Q.......
00000090: A6 59 74 05 83 C3 20 E2 - ED E3 37 26 8B 57 1A 52 .Yt... ...7&.W.R
000000A0: B8 01 00 68 00 20 07 33 - DB 0E E8 48 00 72 28 5B ...h. .3...H.r([
000000B0: 8D 36 0B 00 8D 3E 0B 02 - 1E 8F 45 02 C7 05 F5 00 .6...>....E.....
000000C0: 1E 8F 45 06 C7 45 04 0E - 01 8A 16 24 00 EA 03 00 ..E..E.....$....
000000D0: 00 20 BE 86 01 EB 03 BE - A2 01 E8 09 00 BE C1 01 . ..............
000000E0: E8 03 00 FB EB FE AC 0A - C0 74 09 B4 0E BB 07 00 .........t......
000000F0: CD 10 EB F2 C3 50 4A 4A - A0 0D 00 32 E4 F7 E2 03 .....PJJ...2....
00000100: 06 08 02 83 D2 00 A3 13 - 02 89 16 15 02 58 A2 07 .............X..
00000110: 02 A1 13 02 8B 16 15 02 - 03 06 1C 00 13 16 1E 00 ................
00000120: F7 36 18 00 FE C2 88 16 - 06 02 33 D2 F7 36 1A 00 .6........3..6..
00000130: 88 16 25 00 A3 04 02 A1 - 18 00 2A 06 06 02 40 3A .. % .......*...@:
00000140: 06 07 02 76 05 A0 07 02 - 32 E4 50 B4 02 8B 0E 04 ...v....2.P.....
00000150: 02 C0 E5 06 0A 2E 06 02 - 86 E9 8B 16 24 00 CD 13 ............$...
00000160: 0F 83 05 00 83 C4 02 F9 - CB 58 28 06 07 02 76 11 .........X(...v.
00000170: 01 06 13 02 83 16 15 02 - 00 F7 26 0B 00 03 D8 EB ..........&.....
00000180: 90 A2 07 02 F8 CB 42 4F - 4F 54 3A 20 43 6F 75 6C ......BOOT: Coul
00000190: 64 6E 27 74 20 66 69 6E - 64 20 4E 54 4C 44 52 0D dn't find NTLDR.
000001A0: 0A 00 42 4F 4F 54 3A 20 - 49 2F 4F 20 65 72 72 6F ..BOOT: I/O erro
000001B0: 72 20 72 65 61 64 69 6E - 67 20 64 69 73 6B 0D 0A r reading disk..
000001C0: 00 50 6C 65 61 73 65 20 - 69 6E 73 65 72 74 20 61 .Please insert a
000001D0: 6E 6F 74 68 65 72 20 64 - 69 73 6B 00 4E 54 4C 44 nother disk.NTLD
000001E0: 52 20 20 20 20 20 20 00 - 00 00 00 00 00 00 00 00 R ........ 000001F0:
00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 55 AA ..............U.
```

Tables 1.7 and 1.8 illustrate the layout of the BPB and the extended BPB for FAT16 volumes. The sample values correspond to the data in the preceding example.

Table 1.7 BPB Fields for FAT16 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x0B	WORD	0x0002	**Bytes Per Sector** . The size of a hardware sector. Valid decimal values for this field are 512, 1024, 2048, and 4096. For most disks used in the United States, the value of this field is 512.
0x0D	BYTE	0x40	**Sectors Per Cluster** . The number of sectors in a cluster. Because FAT16 can track only a limited number of clusters (up to 65,536), large volumes are supported by increasing the number of sectors per cluster. The default cluster size for a volume depends on the volume size. Valid decimal values for this field are 1, 2, 4, 8, 16, 32, 64, and 128. Values that lead to clusters larger than 32 KB (**Bytes Per Sector** * **Sectors Per Cluster**) can cause disk and software errors.

0x0E	WORD	0x0100	**Reserved Sectors** . The number of sectors preceding the start of the first FAT, including the boot sector. The value of this field is always 1.
0x10	BYTE	0x02	**Number of FATs** . The number of copies of the FAT on the volume. The value of this field is always 2.
0x11	WORD	0x0002	**Root Entries** . The total number of 32-byte file and folder name entries that can be stored in the root folder of the volume. On a typical hard disk, the value of this field is 512. One entry is always used as a Volume Label, and files and folders with long names use multiple entries per file. The largest number of file and folder entries is typically 511, but entries run out before you reach that number if long file names are used.
0x13	WORD	0x0000	**Small Sectors** . The number of sectors on the volume represented in 16 bits (< 65,536). For volumes larger than 65,536 sectors, this field has a value of zero and the **Large Sectors** field is used instead.
0x15	BYTE	0xF8	**Media Descriptor** . Provides information about the media being used. A value of 0xF8 indicates a hard disk and 0xF0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 disks and are not used in Windows 2000.
0x16	WORD	0xFC00	**Sectors Per FAT** . The number of sectors occupied by each FAT on the volume. The computer uses this number and the number of FATs and hidden sectors, to determine where the root directory begins. The computer can also determine where the user data area of the volume begins based on the number of entries in the root directory (512).
0x18	WORD	0x3F00	**Sectors Per Track** . Part of the apparent disk geometry used on a low-level formatted disk.
0x1A	WORD	0x4000	**Number of Heads** . Part of the apparent disk geometry used on a low-level formatted disk.
0x1C	DWORD	0x3F000000	**Hidden Sectors** . The number of sectors on the volume before the boot sector. This value is used during the boot sequence to calculate the absolute offset to the root directory and data areas.
0x20	DWORD	0x01F03E00	**Large Sectors** . If the value of the **Small Sectors** field is zero, this field contains the total number of sectors in the FAT16 volume. If the value of the **Small Sectors** field is not zero, the value of this field is zero.

Table 1.8 Extended BPB Fields for FAT16 Volumes

Byte Offset	Field Length	Value	Field Name and Definition

0x24	BYTE	0x80	**Physical Drive Number** . Related to the BIOS physical drive number. Floppy disk drives are identified as 0x00 and physical hard disks are identified as 0x80, regardless of the number of physical disk drives. Typically, this value is set prior to issuing an INT 13h BIOS call to specify the device to access. The value is only relevant if the device is a boot device.
0x25	BYTE	0x00	**Reserved** . FAT16 volumes are always set to zero.
0x26	BYTE	0x29	**Extended Boot Signature** . A field that must have the value 0x28 or 0x29 to be recognized by Windows 2000.
0x27	DWORD	0xA88B3652	**Volume Serial Number** . A random serial number created when formatting a disk, which helps to distinguish between disks.
0x2B	11 bytes	NO NAME	**Volume Label** . A field once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x36	LONGLONG	FAT16	**File System Type** . A field with a value of either FAT, FAT12 or FAT16, depending on the disk format.

Top Of Page

FAT32 Boot Sector

Table 1.9 describes the boot sector of a volume formatted with the FAT32 file system.

Note

The FAT32 boot sector is structurally very similar to the FAT16 boot sector, but the FAT32 BPB contains additional fields. The FAT32 extended BPB uses the same fields as FAT16, but the offset addresses of these fields within the boot sector are different than those found in FAT16 boot sectors. Drives formatted in FAT32 are not readable by operating systems that are incompatible with FAT32.

Table 1.9 Boot Sector Sections on a FAT32 Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	53 bytes	BPB
0x40	26 bytes	Extended BPB
0x5A	420 bytes	Bootstrap Code

0x01FE	WORD	End of Sector Marker

The following example illustrates a hexadecimal printout of the boot sector on a FAT32 volume. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B– 0x59 are the BPB and the extended BPB.
- The remaining section is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 878, Side 0, Sector 1

00000000: EB 58 90 4D 53 44 4F 53 - 35 2E 30 00 02 08 20 00 .X.MSDOS5.0 ... .

00000010: 02 00 00 00 00 F8 00 00 - 3F 00 FF 00 EE 39 D7 00 ........?....9..

00000020: 7F 32 4E 00 83 13 00 00 - 00 00 00 00 02 00 00 00 2N............

00000030: 01 00 06 00 00 00 00 00 - 00 00 00 00 00 00 00 00 ................

00000040: 80 00 29 8B 93 6D 54 4E - 4F 20 4E 41 4D 45 20 20 ..)..mTNO NAME

00000050: 20 20 46 41 54 33 32 20 - 20 20 33 C9 8E D1 BC F4 FAT32 3.....

00000060: 7B 8E C1 8E D9 BD 00 7C - 88 4E 02 8A 56 40 B4 08 {......|.N..V@..
00000070: CD 13 73 05 B9 FF FF 8A - F1 66 0F B6 C6 40 66 0F ..s......f...@f.
00000080: B6 D1 80 E2 3F F7 E2 86 - CD C0 ED 06 41 66 0F B7 ....?.......Af..
00000090: C9 66 F7 E1 66 89 46 F8 - 83 7E 16 00 75 38 83 7E .f..f.F..~..u8.~
000000A0: 2A 00 77 32 66 8B 46 1C - 66 83 C0 0C BB 00 80 B9 *.w2f.F.f.......
000000B0: 01 00 E8 2B 00 E9 48 03 - A0 FA 7D B4 7D 8B F0 AC ...+..H...}.}...
000000C0: 84 C0 74 17 3C FF 74 09 - B4 0E BB 07 00 CD 10 EB ..t.<.t.........
000000D0: EE A0 FB 7D EB E5 A0 F9 - 7D EB E0 98 CD 16 CD 19 ...}....}.......
000000E0: 66 60 66 3B 46 F8 0F 82 - 4A 00 66 6A 00 66 50 06 f`f;F...J.fj.fP.
000000F0: 53 66 68 10 00 01 00 80 - 7E 02 00 0F 85 20 00 B4 Sfh.....~.... ..
00000100: 41 BB AA 55 8A 56 40 CD - 13 0F 82 1C 00 81 FB 55 A..U.V@........U
00000110: AA 0F 85 14 00 F6 C1 01 - 0F 84 0D 00 FE 46 02 B4 .............F..
00000120: 42 8A 56 40 8B F4 CD 13 - B0 F9 66 58 66 58 66 58 B.V@......fXfXfX
00000130: 66 58 EB 2A 66 33 D2 66 - 0F B7 4E 18 66 F7 F1 FE fX.*f3.f..N.f...
00000140: C2 8A CA 66 8B D0 66 C1 - EA 10 F7 76 1A 86 D6 8A ...f..f....v....
00000150: 56 40 8A E8 C0 E4 06 0A - CC B8 01 02 CD 13 66 61 V@............fa
00000160: 0F 82 54 FF 81 C3 00 02 - 66 40 49 0F 85 71 FF C3 ..T.....f@I..q..
00000170: 4E 54 4C 44 52 20 20 20 - 20 20 20 0D 0A 4E 54 4C NTLDR ..NTL 00000180:
44 52 20 69 73 20 6D 69 - 73 73 69 6E 67 FF 0D 0A DR is missing... 00000190: 44
69 73 6B 20 65 72 72 - 6F 72 FF 0D 0A 50 72 65 Disk error...Pre 000001A0: 73 73
20 61 6E 79 20 6B - 65 79 20 74 6F 20 72 65 ss any key to re 000001B0: 73 74 61
72 74 0D 0A 00 - 00 00 00 00 00 00 00 00 start..... 000001C0: 00 00 00 00
00 00 00 00 - 00 00 00 00 00 00 00 00 ..... 000001D0: 00 00 00 00 00
```

```
00 00 00 - 00 00 00 00 00 00 00 00 ..............000001E0: 00 00 00 00 00 00
00 00 - 00 00 00 00 00 00 00 00 ..............000001F0: 00 00 00 00 00 00 00 00
00 - 00 7B 8E 9B 00 00 55 AA .........{....U.
```

Tables 1.10 and 1.11 illustrate the layout of the BPB and the extended BPB for FAT32 volumes. The sample values correspond to the data in the preceding example.

Table 1.10 BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x0B	WORD	0x0002	**Bytes Per Sector** . The size of a hardware sector. Valid decimal values for this field are 512, 1024, 2048, and 4096. For most disks used in the United States, the value of this field is 512.
0x0D	BYTE	0x08	**Sectors Per Cluster** . The number of sectors in a cluster. Because FAT32 can only track a finite number of clusters (up to 4,294,967,296), extremely large volumes are supported by increasing the number of sectors per cluster. The default cluster size for a volume depends on the volume size. Valid decimal values for this field are 1, 2, 4, 8, 16, 32, 64, and 128. The Windows 2000 implementation of FAT32 allows for the creation of volumes only up to a maximum of 32 GB. However, larger volumes created by other operating systems (Windows 95 OSR2 and later) are accessible in Windows 2000.
0x0E	WORD	0x0200	**Reserved Sectors** . The number of sectors preceding the start of the first FAT, including the boot sector. The decimal value of this field is typically 32.
0x10	BYTE	0x02	**Number of FATs** . The number of copies of the FAT on the volume. The value of this field is always 2.
0x11	WORD	0x0000	**Root Entries (FAT12/FAT16 only)** . For FAT32 volumes, this field must be set to zero.
0x13	WORD	0x0000	**Small Sectors (FAT12/FAT16 only)** . For FAT32 volumes, this field must be set to zero.
0x15	BYTE	0xF8	**Media Descriptor** . Provides information about the media being used. A value of 0xF8 indicates a hard disk and 0xF0 indicates a high-density 3.5-inch floppy disk. Media descriptor entries are a legacy of MS-DOS FAT16 disks and are not used in Windows 2000.

0x16	WORD	0x0000	**Sectors Per FAT (FAT12/FAT16 only)** . For FAT32 volumes, this field must be set to zero.
0x18	WORD	0x3F00	**Sectors Per Track** . Contains the "sectors per track" geometry value for disks that use INT 13h. The volume is broken down into tracks by multiple heads and cylinders.
0x1A	WORD	0xFF00	**Number of Heads** . Contains the "count of heads" geometry value for disks that use INT 13h. For example, on a 1.44-MB, 3.5-inch floppy disk this value is 2.
0x1C	DWORD	0xEE39D700	**Hidden Sectors** . The number of sectors on the volume before the boot sector. This value is used during the boot sequence to calculate the absolute offset to the root directory and data areas. This field is generally only relevant for media that are visible on interrupt 13h. It must always be zero on media that are not partitioned.
0x20	DWORD	0x7F324E00	**Large Sectors** . Contains the total number of sectors in the FAT32 volume.
0x24	DWORD	0x83130000	**Sectors Per FAT (FAT32 only)** . The number of sectors occupied by each FAT on the volume. The computer uses this number and the number of FATs and hidden sectors (described in this table), to determine where the root directory begins. The computer can also determine where the user data area of the volume begins based on the number of entries in the root directory.
0x28	WORD	0x0000	**Extended Flags (FAT32 only)** . The value of the bits in this two-byte structure are: Bits 0–3: Number of the active FAT (starting count at 0, not 1). It is only valid if mirroring is disabled. Bits 4–6: Reserved. Bit 7: A value of 0 means the FAT is mirrored at run time into all FATs. A value of 1 means only one FAT is active (referenced in bits 0-3). Bits 8–15: Reserved.
0x2A	WORD	0x0000	**File System Version (FAT32 only)** . The high byte is the major revision number, whereas the low byte is the minor revision number. This field supports the ability to extend the FAT32 media type in the future with concern for old FAT32 drivers mounting the volume. If the field is non-zero, back-level Windows

			versions will not mount the volume.
0x2C	DWORD	0x02000000	**Root Cluster Number (FAT32 only)** . The cluster number of the first cluster of the root directory. This value is typically, but not always, 2.
0x30	WORD	0x0100	**File System Information Sector Number (FAT32 only)** . The sector number of the File System Information (FSINFO) structure in the reserved area of the FAT32 volume. The value is typically 1. A copy of the FSINFO structure is kept in the Backup Boot Sector, but it is not kept up-to-date.
0x34	WORD	0x0600	**Backup Boot Sector (FAT32 only)** . A non-zero value indicates the sector number in the reserved area of the volume in which a copy of the boot sector is stored. The value of this field is typically 6. No other value is recommended.
0x36	12 bytes	0x000000000000000000000000	**Reserved (FAT32 only)** . Reserved space for future expansion. The value of this field should always be zero.

Table 1.11 Extended BPB Fields for FAT32 Volumes

Byte Offset	Field Length	Value	Field Name and Definition
0x40	BYTE	0x80	**Physical Drive Number** . Related to the BIOS physical drive number. Floppy disk drives are identified as 0x00 and physical hard disks are identified as 0x80, regardless of the number of physical disk drives. Typically, this value is set prior to issuing an INT 13h BIOS call to specify the device to access. It is only relevant if the device is a boot device.
0x41	BYTE	0x00	**Reserved** . FAT32 volumes are always set to zero.
0x42	BYTE	0x29	**Extended Boot Signature** . A field that must have the value 0x28 or 0x29 to be recognized by Windows 2000.
0x43	DWORD	0xA88B3652	**Volume Serial Number** . A random serial number created when formatting a disk, which helps to distinguish between disks.
0x47	11 bytes	NO NAME	**Volume Label** . A field once used to store the volume label. The volume label is now stored as a special file in the root directory.
0x52	LONGLONG	FAT32	**System ID** . A text field with a value of FAT32.

Top Of Page

NTFS Boot Sector

Table 1.12 describes the boot sector of a volume formatted with NTFS. The bootstrap code for an NTFS volume is longer than the 426 bytes, as shown in Table 1.12. When you format an NTFS volume, the format program allocates the first 16 sectors for the boot sector and the bootstrap code.

Table 1.12 Boot Sector Sections on an NTFS Volume

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B	25 bytes	BPB
0x24	48 bytes	Extended BPB
0x54	426 bytes	Bootstrap Code
0x01FE	WORD	End of Sector Marker

On NTFS volumes, the data fields that follow the BPB form an extended BPB. The data in these fields enables Ntldr to find the master file table (MFT) during startup. On NTFS volumes, the MFT is not located in a predefined sector, as on FAT16 and FAT32 volumes. For this reason, the MFT can be moved if there is a bad sector in its normal location. However, if the data is corrupted, the MFT cannot be located, and Windows 2000 assumes that the volume has not been formatted.

The following example illustrates the boot sector of an NTFS volume formatted while running Windows 2000. The printout is formatted in three sections:

- Bytes 0x00– 0x0A are the jump instruction and the OEM ID (shown in bold print).
- Bytes 0x0B–0x53 are the BPB and the extended BPB.
- The remaining code is the bootstrap code and the end of sector marker (shown in bold print).

```
Physical Sector: Cyl 0, Side 1, Sector 1

00000000: EB 52 90 4E 54 46 53 20 - 20 20 20 00 02 08 00 00 .R.NTFS .....

00000010: 00 00 00 00 00 F8 00 00 - 3F 00 FF 00 3F 00 00 00 ........?...?...

00000020: 00 00 00 00 80 00 80 00 - 4A F5 7F 00 00 00 00 00 ........J.......

00000030: 04 00 00 00 00 00 00 00 - 54 FF 07 00 00 00 00 00 ........T.......

00000040: F6 00 00 00 01 00 00 00 - 14 A5 1B 74 C9 1B 74 1C ...........t..t.
```

```
00000050: 00 00 00 00 FA 33 C0 8E - D0 BC 00 7C FB B8 C0 07 .... .3.....|....

00000060: 8E D8 E8 16 00 B8 00 0D - 8E C0 33 DB C6 06 0E 00 ..........3.....
00000070: 10 E8 53 00 68 00 0D 68 - 6A 02 CB 8A 16 24 00 B4 ..S.h..hj....$..
00000080: 08 CD 13 73 05 B9 FF FF - 8A F1 66 0F B6 C6 40 66 ...s......f...@f
00000090: 0F B6 D1 80 E2 3F F7 E2 - 86 CD C0 ED 06 41 66 0F .....?.......Af.
000000A0: B7 C9 66 F7 E1 66 A3 20 - 00 C3 B4 41 BB AA 55 8A ..f..f. ...A..U.
000000B0: 16 24 00 CD 13 72 0F 81 - FB 55 AA 75 09 F6 C1 01 .$...r...U.u....
000000C0: 74 04 FE 06 14 00 C3 66 - 60 1E 06 66 A1 10 00 66 t......f`..f...f
000000D0: 03 06 1C 00 66 3B 06 20 - 00 0F 82 3A 00 1E 66 6A ....f;. ...:..fj
000000E0: 00 66 50 06 53 66 68 10 - 00 01 00 80 3E 14 00 00 .fP.Sfh.....>...
000000F0: 0F 85 0C 00 E8 B3 FF 80 - 3E 14 00 00 0F 84 61 00 ........>.....a.
00000100: B4 42 8A 16 24 00 16 1F - 8B F4 CD 13 66 58 5B 07 .B..$.......fX[.
00000110: 66 58 66 58 1F EB 2D 66 - 33 D2 66 0F B7 0E 18 00 fXfX.-f3.f.....
00000120: 66 F7 F1 FE C2 8A CA 66 - 8B D0 66 C1 EA 10 F7 36 f......f..f....6
00000130: 1A 00 86 D6 8A 16 24 00 - 8A E8 C0 E4 06 0A CC B8 ......$.........
00000140: 01 02 CD 13 0F 82 19 00 - 8C C0 05 20 00 8E C0 66 ........... ...f
00000150: FF 06 10 00 FF 0E 0E 00 - 0F 85 6F FF 07 1F 66 61 ..........o..fa
00000160: C3 A0 F8 01 E8 09 00 A0 - FB 01 E8 03 00 FB EB FE ................
00000170: B4 01 8B F0 AC 3C 00 74 - 09 B4 0E BB 07 00 CD 10 .....<.t........
00000180: EB F2 C3 0D 0A 41 20 64 - 69 73 6B 20 72 65 61 64 .....A disk read
00000190: 20 65 72 72 6F 72 20 6F - 63 63 75 72 72 65 64 00 error occurred.
000001A0: 0D 0A 4E 54 4C 44 52 20 - 69 73 20 6D 69 73 73 69 ..NTLDR is missi
000001B0: 6E 67 00 0D 0A 4E 54 4C - 44 52 20 69 73 20 63 6F ng...NTLDR is co
000001C0: 6D 70 72 65 73 73 65 64 - 00 0D 0A 50 72 65 73 73 mpressed...Press
000001D0: 20 43 74 72 6C 2B 41 6C - 74 2B 44 65 6C 20 74 6F Ctrl+Alt+Del to
000001E0: 20 72 65 73 74 61 72 74 - 0D 0A 00 00 00 00 00 00 restart........
000001F0: 00 00 00 00 00 00 00 00 - 83 A0 B3 C9 00 00 55 AA ..............U.
```

Table 1.13 describes the fields in the BPB and the extended BPB on NTFS volumes. The fields starting at 0x0B, 0x0D, 0x15, 0x18, 0x1A, and 0x1C match those on FAT16 and FAT32 volumes. The sample values correspond to the data in the preceding example.

Table 1.13 BPB and Extended BPB Fields on NTFS Volumes

Byte Offset	Field Length	Sample Value	Field Name
0x0B	WORD	0x0002	Bytes Per Sector
0x0D	BYTE	0x08	Sectors Per Cluster
0x0E	WORD	0x0000	Reserved Sectors
0x10	3 BYTES	0x000000	*always 0*
0x13	WORD	0x0000	*not used by NTFS*

0x15	BYTE	0xF8	Media Descriptor
0x16	WORD	0x0000	*always 0*
0x18	WORD	0x3F00	Sectors Per Track
0x1A	WORD	0xFF00	Number Of Heads
0x1C	DWORD	0x3F000000	Hidden Sectors
0x20	DWORD	0x00000000	*not used by NTFS*
0x24	DWORD	0x80008000	*not used by NTFS*
0x28	LONGLONG	0x4AF57F0000000000	Total Sectors
0x30	LONGLONG	0x0400000000000000	Logical Cluster Number for the file $MFT
0x38	LONGLONG	0x54FF070000000000	Logical Cluster Number for the file $MFTMirr
0x40	DWORD	0xF6000000	Clusters Per File Record Segment
0x44	DWORD	0x01000000	Clusters Per Index Block
0x48	LONGLONG	0x14A51B74C91B741C	Volume Serial Number
0x50	DWORD	0x00000000	Checksum

Top Of Page

Protecting the Boot Sector

Because a normally functioning system relies on the boot sector to access a volume, it is highly recommended that you run disk scanning tools such as Chkdsk regularly, as well as back up all of your data files to protect against data loss if you lose access to a volume.

Top Of Page