

# The Boot Process

By Mark E. Donaldson

## INTRODUCTION

Most of us take the computer boot process (boot strapping) for granted. The real routine is, we turn our computers on, watch some numbers and letters appear on our monitor, hear some strange beeping sounds and other noises, and wait until our operating system permits us to go to work. This all happens automatically, and there is nothing we are required to do except push the button to start the process in action. Few people really care or understand what is happening. However, this is a very methodical and meaningful period, and what occurs during this time can tell us a lot about the health of the machine. It's when the boot process does not go smoothly, or perhaps not at all, that we wish we understood more about what is happening and what it all means.

This white paper intends to explain the boot process in great detail. From the moment the machine is turned until it is ready to use. This paper will assume that everything is going well and normally. However, some of the problems you encounter will be specifically addressed. The basis of troubleshooting problems lies first knowing what is supposed to happen. When it doesn't, then clues for determining problems are naturally at your finger tips.

The term **boot** comes from the term **bootstrap**. It describes the method whereby the PC becomes operational. Just as one pulls on a large boot by the small strap attached to the back, a PC can load a large operating system program by first loading a small program that can then pull in the operating system. A chain of events begins with the application of power and finally results in an operating computer system with software loaded and running. Each event is called by the event before it and initiates the event after it. A **Glossary of Terminology** may be found in **Appendix "C"** at the end of this paper.

This paper assumes the following:

- This is a cold boot.
- You are booting from the hard disk drive.
- The hard disk drive has been low level formatted, partitioned, and high level formatted.
- The PC is an x86 (CISC) based computer.
- That your operating system may not be DOS. Rather, it may also be Windows 9.x, Windows NT, Windows 2000, or UNIX (either Bourne, Bash, or C shells). Perhaps, the computer may even be set up for multiple operating system boot.

## THE BOOT PROCESS AND TROUBLESHOOTING

Tracing the system boot process might help find the location of a problem if error messages displayed when the problem occurs are examined. If an error message is displayed only by a particular program, one can be quite sure the program in question was at least loaded and partially running. Combine this information with the knowledge of the boot sequence, and it is possible to tell how far along the system's startup procedure is. Error messages displayed during the boot process as well as those during normal system operation can be hard to decipher, but the first step in decoding an error message is to know where the message came from, and what program actually sent or displayed the error message. The following programs are capable of displaying error messages during the boot process:

# The Boot Process

By Mark E. Donaldson

- Motherboard ROM BIOS
- Adapter card ROM BIOS extensions
- Master Partition Boot Sector or Master Boot Record (MBR)
- DOS Volume Boot Sector
- System files (IO.SYS, MSDOS.SYS, or WINBOOT.SYS)
- Device drivers (loaded through CONFIG.SYS or the Windows 9.x Registry SYSTEM.DAT)
- A Shell Program (COMMAND.COM) which can be bypassed in Windows 9.x
- Programs run by AUTOEXEC.BAT (which can be bypassed in Windows 9.x)
- Windows (WIN.COM)

## MS-DOS CORE FILES

Even though we will be considering operating systems other than DOS, it may be helpful to understand a few specifics about the MS-DOS core files. These files are core because they play a major role in the boot process. Other operating systems, such as UNIX, have equivalent files that serve basically the same function.

- **Master Boot Record (MBR)** - During partitioning, no matter what file system is specified, the partitioning software writes a special boot program and partition table to the first sector called the **Master Boot Sector**. Because the term *record* sometimes is often used to mean sector, this sector can also be called the **Master Boot Record (MBR)**. Master Boot Record, or Master Boot Sector, is in the first sector, of first track, on first Disk (CHS 0-0-1). The first 446 bytes (0x000h to 0x1BEh) consists of a special program code called the **Bootstrap Loader**. The Bootstrap Loader is the program that locates the first active or bootable partition on the disk. The next 64 bytes (0x1BEh to 0x1EEh) contains the Master Partition Table. This is the table that lists and identifies all the attributes of the partitions you installed while using the **FDISK** utility. The last 2 bytes (0x1FEh to 0x1FFh) of this 512 byte sector is an identification header, also known as the magic number for the BIOS (0xAA55).

When activated by the system BIOS, the MBR Bootstrap Loader then seeks the DOS Volume (or whatever operating system you have flagged as active or bootable) Boot Record and the DOS Bootstrap Loader program. This information is obtained by the Bootstrap Loader program from the Master Partition table. The DOS (or OS) Bootstrap System Loader is the 446 byte boot program on the Volume Boot Sector on this DOS partition. The DOS Bootstrap loader is always on the first sector, of first track, of the active partition. Normally, unless you are multiple booting with several operating systems, this will be the second 512 bytes (second sector) on the first disk.

Even though the MBR is an essential part of the total boot process, it is not the intent of this paper to endeavor on a detailed explanation of this process. For more information on this subject, refer to my white paper *The Master Boot Record and The Master Partition Table*.

**NOTE:** ALL DRIVES MUST BE PARTITIONED EVEN IF THERE IS ONLY ONE SINGLE PRIMARY DOS PARTITION.

- **IO.SYS** - One of the DOS system files required to boot the machine. The first file loaded from disk during the boot. Contains extensions to the ROM BIOS. Hidden file. First file loaded. Loads and controls device drives. Communicates with BIOS. Must be in root directory.

# The Boot Process

By Mark E. Donaldson

- **MSDOS.SYS** - Hidden file. Contains DOS kernel code or shell. Receives requests from applications and translates them into commands that the IO.SYS can execute through device drivers. Must be in root directory.
- **COMMAND.COM** - Hidden File. Is the users interface to the Operating System. Reads commands from keyboard or mouse. It either executes the command or passes it to the **MSDOS.SYS**. Does not need to be in root directory. If not, the OS must be informed with the **COMSPEC** command in the **CONFIG.SYS**. To read:

**COMSPEC=[DRIVE]:\path\command.com**

- **CONFIG.SYS** - A file that can be created to tell DOS how to configure itself when the machine starts up. Can load device drivers, set the number of DOS buffers, and so on.
- **AUTOEXEC.BAT** - A special batch file that DOS executes at startup. Contains any number of DOS commands that are executed automatically.
- **WINBOOT.SYS** - In Windows 9.x, **IO.SYS** replaces the MS-DOS files **IO.SYS** and **MSDOS.SYS**. This real mode operating system file contains the information needed to start the computer. With it, the computer no longer needs the **CONFIG.SYS** and the **AUTOEXEC.BAT** to start the Windows 9.x operating system. The **CONFIG.SYS** and the **AUTOEXEC.BAT** are preserved for backward compatibility with certain applications and drivers. The Windows 9.x **IO.SYS** file is automatically renamed **WINBOOT.SYS** if the computer starts using DOS.

## THE BOOT PROCESS

When a PC is powered on, a chain reaction of events is begun. This is known as the Bootstrapping, or Boot, Process. After several pre-post events, a set of instructions stored in the ROM BIOS of the motherboard are loaded into memory. These instructions, which are loaded into the last 128 K of RAM base memory, control the system during the boot process, and remain as drivers for various hardware in the system during normal operation. The boot process can thus be divided into three parts or segments:

1. **The BIOS (Basic Input/Output System) Initialization and Pre-Post Events**  
The software interface, or master control program, to all the hardware in the system. With the BIOS, a program easily can access features in the system by calling on a standard BIOS program module instead of talking directly to the device.
2. **The POST (Power On Self Test) and Hardware Initialization**  
The set of routines that tests the motherboard, memory, disk controllers, video adapters, keyboard, and other primary system components. Useful when troubleshooting system failures and problems.
3. **Loading of the Operating System (Initial Program Load or Master Bootstrap Loader)**  
The routine that initiates a search for an operating system on a floppy disk, or hard disk. If an operating system is found, it is loaded into memory and given control of the system.

# The Boot Process

By Mark E. Donaldson

Please bear in mind that even though the boot strapping process itself may be divided into three distinct events or parts, some of them may be occurring simultaneously.

## PART I: THE BIOS (BASIC INPUT/OUTPUT SYSTEM) INITIALIZATION & PRE-POST EVENTS

### 1. Switch Power On

The electrical power to the system is turned on.

### 2. Power Good

The power supply performs a self test. When all voltages and current levels are acceptable, the supply indicates that the power is stable and sends the **Power Good** signal to the motherboard and CPU through the **Power Good Sense Line** (this is the orange wire from the power supply to the motherboard). The time from **Switch-On** to **Power Good** is normally between .1 and .5 seconds. The microprocessor timer chip then receives the Power Good signal which causes it to stop generating a reset signal to the microprocessor.

### 3. CPU Reset

- An electrical signal follows a permanently programmed path to the CPU to clear left over data from the chip's internal memory registers.
- The signal additionally resets the CPU's CS (Code Segment) Register and Instruction Pointer (IP) Register CPU register (Instruction Pointer-IP Register) to the hexadecimal number FFFF:0000 (CS:IP FFFF:0000) in the ROM BIOS. This number tells the CPU the address of the next instruction that should be processed.
- The CPU sends signals over the system bus to make sure that all the components are connected and that they are functioning properly.

### 4. FFFF:0000 JMP

The microprocessor sets itself up in real mode and begins executing the ROM BIOS code starting at memory address FFFF:0000. This location is exactly 16 bytes short of the top of the one megabyte real addressing range, absolute address FFFF:0000 (Hex). This location holds a special **JMP** (jump) instruction that points to another address (typically EA5BE:000F0) where the **BIOS Initialization Code** actually begins (see Figure 1).

### 5. The ROM BIOS Initialization Code And Boot Program Is Invoked By The CPU.

### 6. Low Memory Test

# The Boot Process

By Mark E. Donaldson

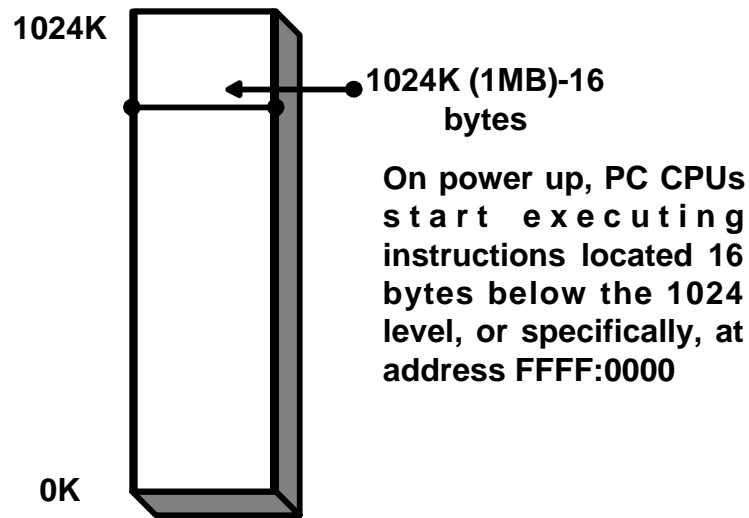


Figure 1

Since the ROM BIOS needs RAM to function, it must first test the bottom part of the systems RAM memory. IF this test fails or crashes, the BIOS cannot recover. This leads to the very first boot problem you may encounter:

**PROBLEM:** One major reason for a completely dead PC is that the lowest band of RAM has failed. If your PC is dead at this point, replace the first bank of memory.

## 7. Hardware Test

The ROM BIOS performs a test of the central hardware to verify basic system functionality. Any errors that occur are indicated by audio codes (see **Appendix "A"**) because the video system has not yet been initialized.

**NOTE:** Often times this portion of the BIOS initialization is mistaken for the computer POST. It is not. The POST routine has not yet been called for by the system BIOS. At this time, the system is merely taking an inventory of its hardware and devices.

## 8. Video ROM Scan

After the system is sure of it's own integrity, the ROM BIOS checks to see if there are installed expansion boards that hold additional BIOS code. The BIOS performs a video ROM scan of memory locations C0000:0000 through C780:0000, looking for video adapter ROM BIOS program contained on a video adapter card plugged into a slot (see Figure 2).

## 9. Video Initialization

If a video ROM BIOS is found, it is tested by a checksum procedure. If it passes the checksum test, the ROM is executed, the ROM code initializes the video adapter, the video resolution and refresh rate are set, and a cursor appears on the screen. If the checksum test fails, the following message appears on the screen:

# The Boot Process

By Mark E. Donaldson

## C000 ROM Error

If the BIOS finds no video adapter ROM, it uses the motherboard ROM video drivers to initialize the video display hardware, and a cursor appears on the screen.

### 10. Scan for Other BIOS

Since the BIOS cannot support every piece of hardware, inventory and analysis may need to be passed off elsewhere. Many extension boards (such as LAN cards and SCSI cards), besides the video board, have their own ROM BIOS with their own initialization code. The BIOS on these add on cards contain three signature bytes to identify themselves to the main BIOS. The first bytes are 55h, then AAh, and then a number indicating how long the BIOS will be (the length of the BIOS divided by 512). The main BIOS, after it locates them, allows the add on boards to perform their inventory and initialization first. The main system BIOS does this by examining memory in the ROM area between 768K and 960K.

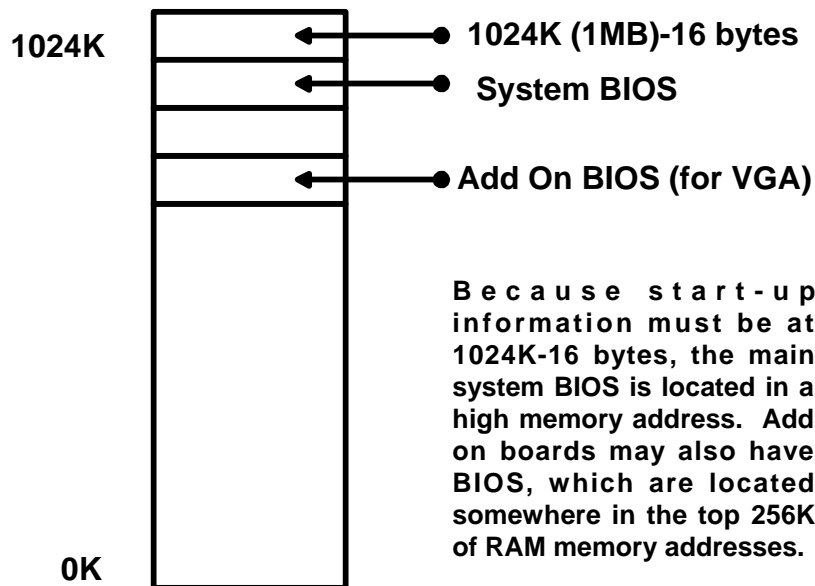


Figure 2

### 11. Expansion Board (Adapter Card) Initialization

The motherboard ROM BIOS scans memory locations C800:0000 through DF80:0000 in 2K increments for any other ROM's located on any other adapter cards. If any ROM's are found, they are checksum tested and executed. These adapter ROM's can alter existing BIOS routines as well as establish new ones. Basically, they reserve the right to override the motherboard ROM BIOS. Failure of a checksum test for any of these ROM modules causes this message to appear on the screen:

## XXXX ROM Error

The address XXX indicates the segment address of the failed ROM module.

# The Boot Process

By Mark E. Donaldson

## 12. ROM BIOS Identification

Information Prints To Screen

## 13. The BIOS Initialization Code

The ROM BIOS invokes the **Power On Self Test** program, or **POST**.

## 14. Hard Drive

The hard drive is also brought up to operating RPM speed while all this is going on.

## PART II: HARDWARE INITIALIZATION (INVENTORY) & POWER ON SELF TEST (POST)

During any part of the hardware initialization and POST process, errors or failures may occur. Errors may be fatal or non-fatal. Fatal errors will result in system lockup, or a failure of the boot process to continue. With non-fatal errors, the boot process will continue, but either an audible or visual error message will occur. Both audible and visual error messages may be given. Visual error will occur only after the video adapter has past POST and the video monitor is readable. Common audible error messages may be found in **Appendix "A"**, while common visual messages are listed in **Appendix "B"**.

### 1. Word Value Check

The ROM BIOS checks the word value at memory location 0000:0472 to see if this is a cold start (boot) or a warm start (boot). A word value of 1234h in this location is a flag that indicates a warm start. A warm start causes the memory test portion of the **POST** to be skipped. Any other word value in the location indicates a cold start and full **POST**.

### 2. Memory Test

If this is a cold start, the actual **POST** program executes. The most visible consequence of the POST is the memory count, or **Memory Test**, where a number appears in the upper left corner of the screen and counts up until it reaches the total amount of memory (less the shadow area) installed on the machine.

### 3. POST

The ROM BIOS POST program performs a test of the central hardware to verify basic system functionality, and to check that all the components are operating properly. The microprocessor, memory, keyboard, mouse, expansion cards, network cards, and hard disks, floppy disks, and CD ROM's are all checked.

- The ROM BIOS checks the keyboard to make sure it is properly attached and to make sure no keys are depressed.
- Signals are sent to any disk drives and listens for a response to determine what drives are available.

**PROBLEM:** This may be another major (or fatal) problem encountered. If the disk drive does not respond immediately, the BIOS does not give up. Rather, it waits for the drive to respond. This is known as the **Time Out Period**. The disk drive indeed may be dead, but the time out and error message must be received before diagnoses can be made.

# The Boot Process

By Mark E. Donaldson

- Any errors found during the POST are reported by a combination of audio and displayed error messages (**Appendix "A" and Appendix "B"**).

## 4. POST Completion

Successful completion of the POST is indicated by a single beep.

## 5. Adapter Card POST

Once the system ROM BIOS has run its POST routine, each adapter card with a BIOS runs its own POST routine. The card manufacturers determine what appears on the screen during the POST process.

## 6. CMOS Setup Information Read

The setup information is read and stored into memory with the BIOS initialization data.

## 7. BIOS (Basic Input/Output System)

After the operating system has taken control of the microprocessor, the ROM BIOS does not rest. Its firmware also includes several sets of routines that programs can call to carry out ordinary functions, such as typing characters on the screen or to a printer, reading keystrokes, and timing events. Because of this basic library, operating system do not need to worry about the small details.

## PART III: LOADING OF THE OPERATING SYSTEM (MASTER BOOT SECTOR INITIAL PROGRAM LOAD)

The following sequence is true for all operating systems up to a point. So, until further notified, and even though DOS is being referred to, (even if you are using Windows 9.x, Windows NT, OS/2, or UNIX), the following wing will apply. When operating system specifics begin to vary, this paper will branch out to those individually.

### FOR ALL COMPUTERS AND OPERATING SYSTEMS:

#### 1. BIOS Bootstrap Loader - Int 19

Once the POST procedure has been completed (by the motherboard ROM BIOS), **Int 19**, the **ROM BIOS Boot Program** or **BIOS Bootstrap Loader**, is loaded into the 640K Base RAM memory.

#### 2. Floppy Drive A: Volume Boot Sector Check

The ROM BIOS **Int 19** program searches for a **Volume Boot Sector** at **cylinder 0, head 0, sector 1 (the very first sector)** on the first floppy disk (**Drive A**). If a bootable, or system, disk is found, this sector is loaded into memory at 0000:7C00 and tested. If a disk is in the drive but the sector cannot be read (a volume boot sector not present, or corrupted if present), the following error message appears and the system stops:

**Non System disk or disk error  
Replace and strike any key when ready**

If no disk is found in the floppy drive A, **Int 19** continues on with the next step.

#### 3. Master Boot Record

# The Boot Process

By Mark E. Donaldson

The ROM BIOS **Int 19** searches for the **Master Boot Record Boot Loader** program code at **cylinder 0, head 0, sector 1 (the very first sector)** of the hard disk, loads it into memory at address 0000:7C00, and tests it for signature.

**The first 466 bytes** of the **Master Boot Record (Master Boot Sector)** is the **Master Bootstrap Loader**, the program that later locates the **first active or bootable DOS (or OS) partition**. The **next 44 bytes** contains the **Master Partition Table**. The **last 2 bytes** of this 512 byte sector is an **identification (signature) header**, or the magic number.

If the last two bytes (the **identification signature header** or magic number) are not equal to 55AAh, **Int 18** (software interrupt 18) is invoked and the following (or similar depending on the BIOS version) error message is displayed:

**No boot sector on fixed disk (Not boot device available)  
strike F1 to retry boot, F2 for setup utility**

**PROBLEM:** Such error messages indicate that the hard disk has not been partitioned by the **DOS FDISK** program, or that the **Master Boot Sector** is **corrupted**.

## 4. Master Bootstrap Loader

If the last two bytes of the **Master Boot Record** are equal to 55AAh, **Int 19** then loads the code found there into memory at location 0000:7C00 and tests it. This small program code is the **Master Bootstrap Loader**. **Int 19** then jumps to the location 0000:7C00 and invokes **Master Bootstrap Loader**. **Int 19** has now completed its job and passes the boot process over to the **Master Bootstrap Loader**.

## 5. Master Partition Table

The **Master Bootstrap Loader** then seeks to locate an **active** or **bootable** partition) in its partition table, the **Master Partition Table**. If such a partition is found, the boot sector of that partition (volume boot sector) is read into memory at location 0000:7C00. The **Master Bootstrap Loader** program then jumps to memory location 0000:7C00. The **Master Bootstrap Loader** then invokes the small operating system loader program from the **Volume Boot Sector**.

**PROBLEM:** If any boot indicator in **the Master Partition Table** is invalid, or if more than one indicates an active partition, or if no active partition entry is found, ROM BASIC is entered through **Int 18**, and the following error message is displayed and the system stops:

**Invalid partition table**

## 6. Volume (Partition) Boot Sector

If an **active** partition is found in the **Master Partition Table**, that partitions boot program in the **Volume Boot Sector** is read into memory at 0000:7C00, and the **Master Bootstrap Loader** jumps to 0000:7C00 with SI pointing to the partition table entry that describes the partition being booted. The Master Bootstrap Loader uses the CHS based **Int 13H AH=02H** call to read the **Volume Boot Sector** of the active partition.

# The Boot Process

By Mark E. Donaldson

**PROBLEM:** If the **Volume Boot Sector** cannot be read successfully from an active partition within five retries because of read errors, the following error message appears on the screen and the system stops:

## Error loading operating system

The **Volume Boot Sector** is normally at *cylinder 0, head 0, sector 1 (the very first sector)* of the **active** partition. However, each operating system has its own boot sector format. The **Master Bootstrap Loader** must locate the small OS loader program in the **Volume Boot Sector**, and it must locate the first part of the operating system's kernel loader program (or perhaps the kernel itself).

The **DOS Volume Bootstrap Loader** resides within the first 466 bytes of the 512 byte (second 512 bytes) **active DOS Volume Boot Sector**. The *next 2 bytes* of this 512 byte sector is an **identification (signature) header**. The *last 44 bytes* contains the **DOS Volume Boot Table**. The hard disk **DOS Volume Boot Sector** is tested for a signature. If the **DOS Volume Boot Sector** does not contain a valid signature of 55AAh as the last two bytes in the sector, the following error message appears and the system stops:

## Missing operating system

If the **DOS Volume Boot Sector** does contain a valid signature of 55AAh, the **DOS Volume Bootstrap Loader** program is executed.

**NOTE:** **Int 19** is also called when the **CTRL-ALT-DEL** keys are used. On most systems, **CTRL-ALT-DEL** causes a short version of the **POST** to be executed before **Int 19** is called.

**NOTE:** The **Volume Boot Sector** is often referred to as the **Partition Boot Sector**. These are one and the same thing.

## FOR DOS, WINDOWS 3.x, AND WINDOWS 9.X ONLY:

### 7. DOS Volume Bootstrap Loader

The **DOS Volume Bootstrap Loader** checks the root directory to ensure that the first two files are **IO.SYS** and **MSDOS.SYS**, or **WINBOOT.SYS**. If these two files are not found, or if there are problems loading them, or if the **DOS Volume Boot Sector** is corrupted, the following error message will occur on the screen and the system stops:

## Non-system disk or disk error Replace and press any key when ready

If no problems occur, the **DOS Volume Bootstrap Loader** executes **IO.SYS** or **WINBOOT.SYS**.

### 8. IO.SYS

The initialization code in **IO.SYS** copies itself into the highest region of contiguous DOS memory and transfers control to the copy. The initialization code copy then relocates **IO.SYS** over the portion of **IO.SYS** in low memory that contains the initialization code, because the initialization

# The Boot Process

By Mark E. Donaldson

code no longer needs to be in that location. **WINBOOT.SYS** combines the functions of **IO.SYS** and **MSDOS.SYS** in Windows 9.x.

The initialization code executes **IO.SYS**, which initializes the base device drivers, determines equipment status, resets the disk system, resets and initializes attached devices, and sets the system default parameters.

## 9. **MSDOS.SYS**

The **IO.SYS** executes **MSDOS.SYS**, which is loaded into the first 640K of memory. The full DOS filing system is active at his time, and the **IO.SYS** initialization code is given back control.

## 10. **COMMAND.COM**

The **MSDOS.SYS** executes the **COMMAND.COM** and loads it into the first 640K of base memory. In Windows 9.x, the **COMMAND.COM** program is loaded only if an **AUTOEXEC.BAT** file exists, so that it can process the commands contained within.

## 11. **CONFIG.SYS**

The **IO.SYS** initialization code reads the **CONFIG.SYS** multiple times, and loads it into the first 640K of base memory. The **WINBOOT.SYS** would look for the **SYSTEM.DAT** registry file.

- **DEVICE** statements are processed first in the order that they appear, and any device driver files named are loaded and executed.
- Any **INSTALL** statements are processed in the order in which they appear, and the programs named are loaded and executed.
- The **SHELL** statement is processed and loads the specified command processor with the specified parameters. If the **CONFIG.SYS** contains no **SHELL** statement, the default **COMMAND.COM** processor is loaded with default parameters. Loading the command processor overwrites the initialization code in memory because the job of the initialization code is over.

## 12. **COMMAND.COM** and **AUTOEXEC.BAT**

If **AUTOEXEC.BAT** is present, **COMMAND.COM** loads and runs **AUTOEXEC.BAT**. After the commands in **AUTOEXEC.BAT** have been executed, the DOS prompt appears and the operating system has taken control.

## 13. **WINBOOT.SYS**

In Windows 9.x, **WINBOOT.SYS** automatically loads **HIMEM.SYS**, **IFSHLP.SYS**, and **SETVER.EXE**. Finally, it loads **WIN.COM** and Windows 9.x is officially started and the operating system has taken control.

## FOR WINDOWS NT ONLY:

### 7. **NTLDR (NT Loader) Invoked**

The **Master Bootstrap Loader** seeks to locate an **active** NTFS in its partition table. If such a partition is found, the boot sector of that partition (partition boot sector) is read into memory at location 0000:7C00. The **Master Bootstrap Loader** program then jumps to memory location

# The Boot Process

By Mark E. Donaldson

0000:7C00. The **Master Bootstrap Loader** then invokes the **NTLDR**, the small operating system loader program from the **Partition Boot Sector**.

**NTLDR** accomplishes the following:

- controls the operating system selection process.
- performs hardware detection prior to the Windows NT Kernel initialization.
- loads the operating system files from the boot partition.

**NTLDR** must be in the root folder of the startup disk and requires that the following files be located in the root folder:

- Ntdetect.com
- Boot.ini
- Bootsect.dos (if dual booting)
- Ntbootdd.sys (if boot disk is SCSI).

## 8. **NTLDR Executed**

The **NTLDR** starts executing. It first clears the screen and displays the boot loader message:

### **OS Loader Vx.0**

and then performs the following steps:

- Switches the CPU into 32 bit flat (protected) memory mode.
- Starts the appropriate minifield system. The code to access files on FAT and NTFS volumes is built into **NTLDR**. This code enables **NTLDR** to read, access, and copy files.
- Reads the Boot.ini files and displays the operating system selections. This screen is referred to as the boot loader screen.

## 9. **Boot Loader Screen (Boot.ini)**

**NTLDR** displays a menu from which to select an operating system. This screen is based upon the information in the Boot.ini file. The screen looks as such (example):

### **OS Loader Vx.0**

**Please select the operating system to start:**

**Windows NT Version x.0**  
**Windows NT Version x.) (VGA mode)**  
**Windows 9.x**  
**MS-DOS**

**Use ↑ and ↓ to move the highlight to your choice.**

# The Boot Process

By Mark E. Donaldson

**Press Enter to choose.**

**Seconds until highlighted choice will be started automatically: 29**

If a selection is not made before the counter reaches zero, **NTLDR** loads the operating system specified by the default parameter in the **Boot.ini** file.

## 10. Hardware Configuration

Once the operating system has been selected, and it is assumed that Windows NT has been selected, and the boot loader has collected hardware information, the following message is displayed on the screen:

**OS Loader Vx.0**

**Press spacebar now to invoke Hardware Profile/Last Known Good Menu**

The boot loader waits a few seconds for the **SPACEBAR** to be depressed. It then loads Windows NT by using the Default control set.

## 11. Kernel Load (Ntoskrnl.exe)

The kernel load phase is entered. Several dots appear on the screen as the boot loader loads the Windows NT Kernel **Ntoskrnl.exe** and the hardware adaptation layer **Hal.dll** into memory. However, the kernel is not yet initialized at this time.

## 12. Registry Load

The boot loader loads the Registry key **HKEY\_LOCAL\_MACHINE\SYSTEM** from **%systemroot\System32\Config\System**. At this point, the boot loader creates the control set it will use to initialize the computer. It scans all of the services in the Registry subkey **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services** for device drivers with a start value of **0x0**, which indicates they should be loaded but not initialized.

The Registry subkey

**KEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder** defines the loading order. The loading of these device drivers into memory is done using BIOS **Int 13** calls in real mode, or by **Ntbootdd.sys**.

## 13. Kernel Initialization

When the Kernel starts initializing, the screen turns blue and a similar following message is displayed on the screen:

**Microsoft ® Windows NT™ Version 4.0 (Build 1345)  
1 System Processor (32 MB Memory)**

This means **Ntoskrnl.exe** has successfully initialized and that control has been passed to it.

## 14. Load and Initialize Device Drivers

# The Boot Process

By Mark E. Donaldson

The Kernel now initializes the low level device drivers that were loaded during the Kernel load phase. If an error occurs, the action taken is based on the **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DriverName\ErrorControl** value for the device driver that has a problem.

## 15. Registry Scan

**Ntoskrnl.exe** scans the Registry for device drivers that have a **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DriverName\Start** value of 0x1. As in the Kernel load phase, the Group value for each device driver determines the order in which they are loaded. The Registry subkey **KEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceGroupOrder** defines the loading order.

## 16. Load Services

The Session Manager **Smss.exe** starts the higher-order subsystems and services for Windows NT. Information for the Session Manager is in **KEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager**. Session Manager executes the instruction under:

- BootExecute data item.
- Memory Management key.
- DOS Devices key.
- Subsystem key.

## 17. BootExecute Data Item

The **BootExecute** data item contains one or more commands that Session Manager runs before it loads any services. The default value for this item is **Autochk.exe**, which is in the Windows NT version of **Chkdsk.exe**. After Session Manager runs the commands, the Kernel loads the other Registry keys from **%systemroot\System32\Config**.

## 18. Memory Management Key

The Session Manager creates the paging information required by the Virtual Memory Manager. The configuration information is located in the data items:

**PagePoolSize : REG\_DWORDF 0**  
**NonPagePoolSize : REG\_DWORD 0**  
**PagingFiles : REG\_MULTI\_SZ : c:\pagefile.sys 32**

## 19. DOS Devices Key

The Session Manager creates symbolic links. These links direct certain classes of commands to the correct component in the file system. The configuration information for these default items is located in:

**PRN : REG\_SZ : \DosDevices\LPT1**  
**AUX : REG\_SZ : \DosDevices\COM1**  
**NUL : REG\_SZ : Device\Null1**  
**UNC : REG\_SZ : \Device\Mup**

# The Boot Process

By Mark E. Donaldson

**PIPE : REG\_SZ : \Device\NamedPipe**

**MAILSLOT : REG\_SZ : \Device\MailSlot**

## 20. Subsystem Key

The Windows subsystem Win32 is started. This subsystem controls all I/O and access to the video screen. The process name for the subsystem is **CSRSS**. The Windows subsystem then starts the **WinLogon** process, which then starts several other vital subsystems.

The configuration information for required subsystems is defined by the value for Required in the Registry subkey

**KEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\Subsystems.**

## 21. WinLogon

The Windows subsystem starts **WinLogon.exe**, and **WinLogon.exe** starts the Local Security Administration **Lsass.exe**. The **Begin Logon** dialog box is displayed on the screen. This contains the text:

**Press Ctrl+Alt+Delete to log on**

At this time, Windows NT might still be initializing network device drivers, but it has cleared the system for log on.

## 22. Service Controller (Screg.exe)

The Service Controller **Screg.exe** executes. It makes a final pass through the Registry looking for services that are marked to load automatically. Auto-load services have a start value of 0x2 in the subkeys **HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DriverName\**. The services that are loaded during this phase are loaded based on their dependencies because they are loaded in parallel.

## FOR UNIX ONLY:

### 7. OS Loader

The **Master Bootstrap Loader** finds and loads the OS Loader within the first 466 bytes of the **UNIX System Partition Boot Sector**. With Linux, this loader is known as LILO, or the Linux Loader.

### 8. Kernel Load

The OS Loader locates and loads the UNIX kernel into memory. After the kernel initializes itself and determines the computer's configuration, it creates a process with A PID of 1 which runs the **/etc/init** program.

### 9. Process #1: /etc/init

**/etc/init** executes the shell script **/etc/rc**.

### 10. /etc/rc

Depending on which version of UNIX being used, **/etc/rc** may execute a variety of other shell scripts whose name begin with **/etc/rc** (common varieties include **/etc/rc.network** or **/etc/rc.local**). When **/etc/rc** finishes executing, **/etc/init** forks a new process for every enabled

# The Boot Process

By Mark E. Donaldson

terminal on the system. On older systems, this program is called **/etc/getty**. On newer systems, this program is called **/usr/lib/saf/ttymon**.

## 11. login:

The **getty** or **tytmon** program displays the word **login**:. When it gets a user-name, **getty/ttymon** exec's the program **/bin/login**.

## 12. /etc/passwd

**/bin/login** requests a password and validates it against the password stored in the **/etc/passwd** file. If the username and password match, the **/bin/login** program performs some accounting and initialization tasks, then changes its real and effective UIDs to be those of the username that has been supplied. **/bin/login** then exec's the users shell program, usually **/bin/csh**, **/bin ksh**, **/bin/sh**, or **/bin/bash**.

## 13. User Shell Execution

After log in, UNIX starts up the users shell. The shell will then read a series of start-up commands from a variety of different files. If the shell is **/bin ksh**, **/bin/sh**, or **/bin/bash**, UNIX will execute all the commands stored in a special file called **.profile** in the users home directory. On some systems, **/bin ksh**, **/bin/sh**, or **/bin/bash**, will also execute the commands stored in the **/etc/profile** or **/usr/lib/profile** file.

If the shell is **/bin/csh**, UNIX will execute all of the commands stored in the **.cshrc** file in the users home directory. The C shell will then execute all the of the commands stored in the **.login** file in the users home directory.

# The Boot Process

By Mark E. Donaldson

## APPENDIX A: AUDIBLE ERROR MESSAGES AMI BIOS

TYPICAL POST AUDIBLE (FATAL ERROR) MESSAGES AMI BIOS	
Message	Meaning
1 beep	DRAM refresh failure. The memory refresh circuitry on the motherboard is faulty.
2 beeps	Parity Circuit failure. A parity error was detected in the base memory (first 64K block) of the system.
3 beeps	Base 64K RAM failure. A memory failure occurred within the first 64K of memory.
4 beeps	System Timer failure. Timer #1 on the system board has failed to function properly.
5 beeps	Processor failure. The CPU on the system board has generated an error.
6 beeps	Keyboard Controller 8042 Gate A20 error. The keyboard controller (8042) contains the gate A20 switch which allows the computer to operate in virtual mode. This error message means that the BIOS is not able to switch the CPU into protected mode.
7 beeps	Virtual Mode (CPU) Exception error. The CPU on the motherboard has generated an Interrupt Failure exception interrupt.
8 beeps	Display memory R/W Test failure. The system video adapter is either missing or its memory is faulty. This is not a fatal error.
9 beeps	ROM BIOS Checksum failure. The ROM checksum value does not match the value encoded in the BIOS. This is a good indication the BIOS ROM has failed.
10 beeps	CMOS Shutdown Register. The shutdown register for the CMOS memory R/W Error has failed.
11 beeps	Cache Error. The external cache is faulty.
No beeps	Power supply is bad or not plugged in.
Continuous beep	Power supply bad or keyboard stuck.
Repeating short beep	Power supply bad.
2 short beeps	POST failed. This is caused by a failure of one of the hardware testing procedures.
1 short beep - blank screen	Display adapter failure, or missing.
1 long - 1 short beep	System board failure.
1 long - 2 short beeps	Video failure. This is caused by one of two possible hardware faults. Video ROM BIOS failure with checksum error, or the video adapter has a horizontal retrace failure.
1 long - 3 short beeps	Video failure. This is caused by one of three possible hardware faults. The video DAC has failed, the monitor detection process has failed, or the video RAM has failed.
1 long beep	POST is successful. This is the one you want. It indicated that all hardware tests were successfully completed without error.

# The Boot Process

By Mark E. Donaldson

## APPENDIX B: VISUAL ERROR MESSAGES AMI BIOS

TYPICAL POST VISUAL (FATAL ERROR) MESSAGES AMI BIOS	
Message	Meaning
8042 Gate - A20 Error	Gate A20 on the keyboard controller (8042) is not working.
Address Line Short	Error in address decoding circuitry.
Cache Memory Bad-Do Not Enable Cache	The cache memory is defective.
Ch-2 Timer Error	There is an error in timer 2. Several systems have two timers.
CMOS Battery State Low	The battery power is getting low.
CMOS Checksum Failure	After CMOS RAM values are saved, a checksum value is generated for error checking. The previous value is different from the current value.
CMOS System Options Not Set	The values stored in CMOS RAM are either corrupt or nonexistent.
CMOS Display Type Mismatch	The video type in CMOS RAM is not the one detected by the BIOS.
CMOS Memory Size Mismatch	The physical amount of memory on the motherboard is different than the amount in CMOS RAM.
CMOS Time and Date Not Set	No time and date.
Diskette Boot Failure	The boot disk in floppy drive A: is corrupted (perhaps by a virus). Possibly indicate that an operating system is not available.
Display Switch Not Proper	A video switch on the motherboard must be set to either color or monochrome.
DMA Error	Error in DMA (Direct Memory Access) controller.
DMA #1 Error	Error in the first DMA channel.
DMA #2 Error	Error in the second DMA channel.
FDD Controller Failure	The BIOS cannot communicate with the floppy disk drive controller.
HDD Controller Failure	The BIOS cannot communicate with the hard disk drive controller.
INTR #1 Error	Interrupt channel 1 failed POST.
INTR #2 Error	Interrupt channel 2 failed POST.
Keyboard Error	There is a timing problem with the keyboard.
KB/Interface Error	There is an error in the keyboard connector.
Parity Error ?????	Parity error in system memory at an unknown address.
Memory Parity Error at xxxxx	Memory failed at the xxxxx address.
I/O Card Parity Error at xxxxx	An expansion card failed at the xxxxx address.
DMA Bus Time Out	Device has used bus signal for more than allocated time.

# The Boot Process

By Mark E. Donaldson

## APPENDIX C: GLOSSARY OF TERMINOLOGY

**Allocation Unit** - Same as Cluster.

**Boot** - The process by which the PC becomes operational, and the loading of the operating system into RAM. The term comes from the phrase *pulling a boot on by the bootstrap*.

**Bootstrap** - A technique or device designed to bring itself into a desired state by means of its own action...The term bootstrap is used to describe the process by which a device such as a PC goes from its initial power on condition to a running condition without human intervention.

**Cluster** - Also called **allocation unit**. A group of sectors on a disk that forms a fundamental unit of storage to the operating system.. Clusters or allocation units size is determined by DOS when the disk is high level formatted. There are one or more sectors on a track.

**Cylinder** - The set of tracks on a disk that are on each side of all of the platters in a stack and are the same distance from the center of the disk. The total number of tracks that can be read without moving the heads. A floppy drive with two heads usually has 160 tracks, which are accessible as 80 cylinders. A typical 4G hard disk has 10 platters with 20 heads (19 for data and one servo head) and 4,000 cylinders, in which each cylinder is composed of 19 tracks.

**Extended Partition** - A nonbootable DOS partition containing DOS volumes. Starting with DOS V3.3, the DOS **FDISK** program can create two partitions that serve DOS. One ordinary bootable partition (called the primary partition) and an extended partition, which may contain as many as 23 volumes from D: through Z:.

**FDISK** - The name of the disk partitioning program under several operating systems used to create the Master Boot Record and allocate partitions for the operating system's use.

**FORMAT** - The DOS format program that performs both low and high level formatting on floppy disks but only high level formatting on hard disks.

**Logical Drive** - A drive as named by a DOS drive specifier , such as C: or D:.. Under DOS 3.3 or later, a single physical drive can act as several logical drives, each with its own specifier. Same as Volume. Anything that DOS assigns a drive letter to in an extended DOS partition.

**Master Boot Record (MBR)** - Same as Master Partition Boot Record and Master Boot Table. Always found on the first sector, of the first track, of the first surface (cylinder 0, head 0 sector 1). A one sector record that tells the computer's built in operating system (BIOS) of the most fundamental facts about a disk and the installed operating systems. Instructs the computer how to load the operating system files into memory, thus booting the machine. The MBR is generally operating system independent.

**Master Boot Sector** - The first sector on the first disk on any system (cylinder 0, head 0 sector 1). The **Master Boot Sector** contains the **Master Boot Record (MBR)** and **Master Partition Table**, or the critical information needed to boot the system.

# The Boot Process

By Mark E. Donaldson

**Partition** A section of a hard disk devoted to a particular operating system. Most hard disks have only one partition which is devoted to DOS. A hard disk can have as many as four partitions, each occupied by a different operating system. DOS V3.3 or higher can occupy two of these four partitions.

**Partition Boot Sector** - Same as **Volume Boot Sector**. The first sector of any bootable or active partition. Contains the code necessary to load the operating system. This is the sector the **Master Bootstrap Loader** finds to invoke the small operating system loader program. The **Partition Boot Sector** is dependent on both the operating system and the file system. Typically it is one sector long. However, on an NTFS volume, it can be up to 16 sectors long.

**Partitioning** - The dividing of a hard disk in to individual segments called **partitions**. When a disk is partitioned, the information is recorded in a record called the **Master Boot Record (MBR)** and **Master Partition Table**. DOS allows for up to four partition tables, or partitions, with three of them being designated as Primary Partitions, and one of them as an Extended Partition. **Primary Partitions** are used for the placement of separate operating systems. **Logical Drives** can only be placed in the Extended Partition.

**Primary Partition** - An ordinary, single volume bootable partition.

**Sector** - The smallest unit into which a track is divided during the low level formatting process. A sector consists of 512 bytes of data space. A section of one track defined with identification markings and an identification number. Most sectors hold 512 bytes.

**Track** - One of many concentric circles that holds data on a disk surface. Consists of a single line of magnetic flux changes and is divided into some number of 513 byte sectors.

**Volume** - Same as a Logical Drive. Anything that DOS assigns a drive letter to in an extended DOS partition. A portion of a disk signified by a single drive specifier. Under DOS V3.3 and later, a single hard disk can be partitioned into several volumes, each with its own logical drive specifier (C:, D:, and so on).

**Volume Boot Sector (VBS)** - Same as **Partition Boot Sector**. The first sector of any bootable or active partition. Contains the code necessary to load the operating system. This is the sector the **Master Bootstrap Loader** finds to invoke the small operating system loader program. The **Partition Boot Sector** is dependent on both the operating system and the file system. Typically it is one sector long. However, on an NTFS volume, it can be up to 16 sectors long.