

# Inside the Blue Screen

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

The color blue has become synonymous with disaster in the Windows NT world. Although NT is more reliable and stable than its cousins, Windows 3.x and Windows 95, it nevertheless is subject to the frailties of third-party software, add-on peripherals and their device drivers, and Microsoft's bugs. Almost everyone who has used NT for any length of time has seen a blue screen (also known as the blue screen of death). Screen 1, page 58, displays a typical example. NT stops processing and paints one of these displays whenever it has encountered a situation in which it cannot continue, or in which continuing may lead to data corruption.

```
*** STOP: 0x000000A (0x00000000,0x00000002,0x00000000,8038c240)
IRQL_NOT_LESS_OR_EQUAL*** Address 8038c240 has base at 8038c000 - Ntfs.SYS

CPUID:Genuine Intel 6.3.3 irql:1f SYSVER 0xf0000565

Dll Base DateStmp - Name
80100000 336546bf - ntoskrnl.exe
80000100 334d3a53 - atapi.sys
802aa000 33013e6b - epst.mpd
802b9000 336015af - CLASS2.SYS
802bd000 33d844be - Siwvid.sys
f9318000 31ec6c8d - Floppy.SYS
f9468000 31ed868b - KSecDD.SYS
f9358000 335bc82a - i8042prt.sys
f947c000 31ec6c94 - kbdclass.sys
f9370000 33248011 - VIDEOPORT.SYS
f9490000 31ec6c6d - vga.sys
f90f0000 332480d0 - Npfs.SYS
a0000000 335157ac - win32k.sys
fe0c9000 335bd30e - Fastfat.SYS
fe108000 31ec6c9b - Parallel.SYS
f9050000 332480ab - Serial.SYS

Dll Base DateStmp - Name
80010000 33247f88 - hal.dll
80007000 33248043 - SCSIIPORT.SYS
802b5000 336016a2 - Disk.sys
8038c000 3356d637 - Ntfs.sys
803e4000 33d84553 - NTice.sys
f95c9000 31ec6c99 - Null.SYS
f95ca000 335e60cf - Beep.SYS
f9474000 3324806f - mouclass.sys
f95cb000 3373c39d - ctrl2cap.SYS
fe9d7000 3370e7b9 - ati.sys
f93b0000 332480dd - Msfs.SYS
fe957000 3356da41 - NDIS.SYS
fe914000 334eal44 - ati.dll
fe110000 31ec7c9b - Parport.SYS
f95b4000 31ec6c9d - ParVdm.SYS

Address dword dump Build [1314] - Name
801afc24 80149905 80149905 ff8e6b8c 80129c2c ff8e6b94 8025c000 - Ntfs.SYS
801afc2c 80129c2c 80129c2c ff8e6b94 00000000 ff8e6b94 80100000 - ntoskrnl.exe
801afc34 801240f2 80124f02 ff8e6df4 ff8e6f60 ff8e6c58 80100000 - ntoskrnl.exe
801afc54 80124f16 80124f16 ff8e6f60 ff8e6c3c 8015ac7e 80100000 - ntoskrnl.exe
801afc64 8015ac7e 8015ac7e ff8e6df4 ff8e6f60 ff8e6c58 80100000 - ntoskrnl.exe
801afc70 80129bda 80129bda 00000000 80088000 80106fc0 80100000 - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option. If this message reappears,
contact your system administrator or technical support group.
```

What most users and many developers don't know is what the screen's information means. If you're lucky, simply resetting the computer will get you on your way. If you're unlucky, you'll repeatedly get a blue screen every time you start NT or perform a particular operation (e.g., inserting a new floppy). Even if you've successfully moved past a blue screen with a reboot, understanding the clues it provides can help you avoid future blue screens or give you a hint about what driver or piece of hardware is causing problems.

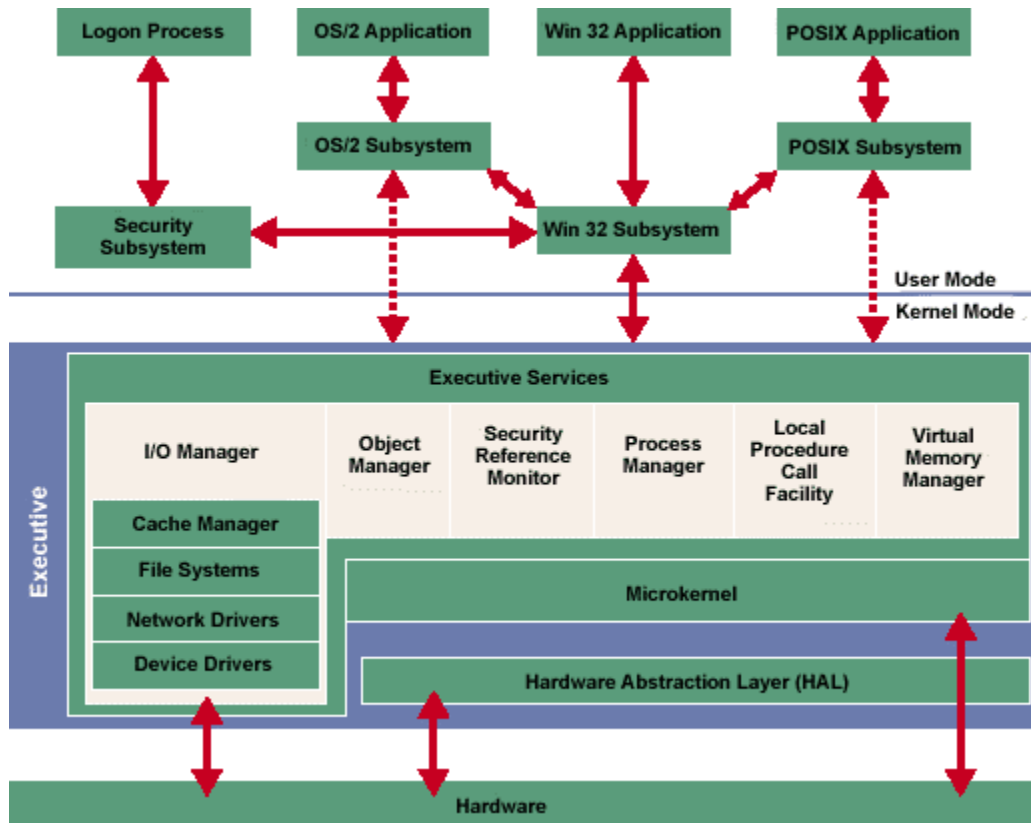
This month, I'll talk about how NT generates blue screens, what leads to their appearance, how to interpret the cryptic data NT lists on them, and how to go about troubleshooting them. I'll tackle the topic from the perspective that NT device drivers are not your forte and that debugging a blue screen with dump analysis tools or a kernel-mode debugger is infeasible. In the process, I'll describe the inner workings of NT's kernel mode.

# Inside the Blue Screen

Mark Russinovich  
(Reprinted from WindowsItPro Magazine)

## NT Architecture Basics

To understand what leads to a blue screen, you first need to understand NT's basic architecture. NT executes in two modes, user mode and kernel mode, as shown in Figure 1, page 59. Kernel mode is a highly privileged processor mode, with direct access to all hardware and memory; user mode is a less privileged mode, with no direct access to hardware and restricted access to memory.



User mode is the mode in which applications and operating system environment subsystems execute. The operating system environments that NT supplies include POSIX, OS/2, Win16, DOS, and Win32. Applications are clients of exactly one environment subsystem and use only the APIs that subsystem exports. Thus, Win32 programs are clients of the Win32 subsystem and use only the Win32 API.

The subsystems use basic NT services that the NT Executive and the Microkernel provide. These services run in kernel mode. The Executive includes core operating system components: the Process Manager, Virtual Memory Manager, I/O Manager, Local Procedure Call (LPC) Facility, Object Manager, and Security Reference Monitor. The Executive is generally portable across processor architectures (e.g., Alpha, x86), and it relies on the Microkernel for processor-specific functions such as context-switching (scheduling) and synchronization primitives.

Beneath the Microkernel resides the Hardware Abstraction Layer (HAL), through which the Executive subsystems and the Microkernel interface with the processor. Microsoft ships different HALs for different processors and processor boards.

Device drivers are modules that interface NT and applications to specific hardware devices. A large number of device drivers for disk drives, video cards, modems, network cards, and input devices ship

# Inside the Blue Screen

Mark Russinovich

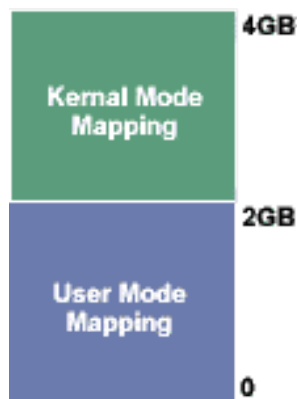
(Reprinted from WindowsItPro Magazine)

with NT. However, hardware vendors can include custom device drivers with their hardware, and NT dynamically adds the drivers to its kernel-mode environment.

## User Mode vs. Kernel Mode

What differentiates user mode from kernel mode is the privilege level. A program executing in user mode runs in a sandbox (not unlike a Java virtual machine's sandbox) that the NT Executive and the program's operating system environment create for the program. The sandbox enforces restrictions as to what the program can do. One type of restriction relates to what parts of the computer's memory the program can reference and in what ways.

Figure 2 shows the virtual memory map that NT creates for applications. Addressable memory totals 4GB, but NT evenly divides the space between the memory assigned to a program and the memory that the kernel-mode portion of NT uses.



The lower 2GB mapping changes, depending on which program is currently running. For example, if Microsoft Word is running, NT places Word's address mapping in the lower 2GB; if Netscape Navigator runs next, its mapping replaces Word's mapping.

The upper 2GB mapping always remains that of the Executive, Microkernel, device drivers, and HAL. Thus, the split between user mode and kernel mode also shows up in NT's address space mapping. (In NT Server 4.0, Enterprise Edition, you can adjust the address split between user mode and kernel mode so that applications have 3GB of memory, with 1GB left for NT's Executive, drivers, and HAL. You will see this split only when NT is running on systems with several gigabytes of physical memory.)

The primary memory restriction placed on user-mode programs is that they cannot access any of the kernel-mode memory. User-mode programs also cannot access invalid portions of their mapping (i.e., portions not filled with data or code from the program). This arrangement contrasts with the kernel-mode portions of NT, which have free rein over the entire address map. For example, NT does not stop a device driver from writing data into Word's address map, but NT prevents Word from writing over the device driver's image.

The user-mode sandbox enforces another restriction that limits a program's ability to directly access hardware devices such as disks, the video screen, and the printer. Programs must typically go through their operating system environment (e.g., Win32) to read data from or write data to a peripheral. The operating system environment then usually calls on the services of the Executive in kernel mode, effectively forwarding the request. The Executive finally completes the request, sometimes with the aid of a device driver, but almost always with the use of functions in the HAL that

# Inside the Blue Screen

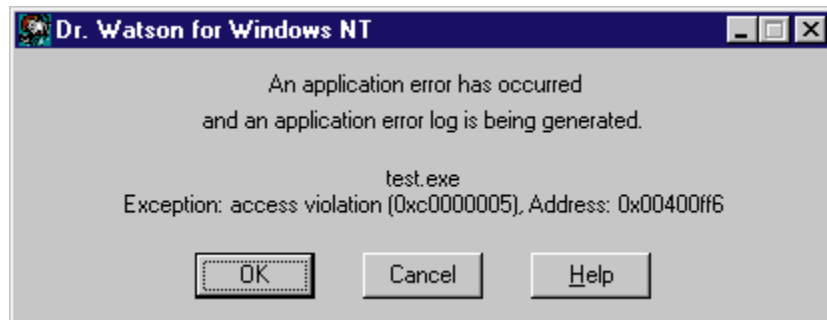
Mark Russinovich

(Reprinted from WindowsItPro Magazine)

interface with the computer's hardware. NT implements the transition between user mode and kernel mode as a system call gateway, through which the passage of data is precisely controlled.

Although a user-mode program can try to directly communicate with a hardware device, NT prevents it from doing so. Any kernel-mode component, however, can touch any part of the hardware. For example, a device driver implemented to interface with a disk drive can access video hardware without NT stopping it.

What do I mean when I say that NT stops user-mode programs from reaching outside their sandboxes to touch memory that isn't theirs or access hardware devices directly? You probably have seen the result of such an attempt, which Screen 2 shows. The infamous Dr. Watson dialog box signals that NT caught a program doing something illegal, and NT is terminating the program. The detection of such transgressions takes place in a kernel-mode subsystem such as the Process Manager or the Virtual Memory Manager. Some legal user-mode operations (e.g., referencing memory that the paging file is currently using) generate processor exceptions, but a program can also trigger exceptions when it steps outside its sandbox. A kernel-mode component must determine whether an exception is the result of a legal or an illegal operation; when a kernel-mode component catches an illegal exception, it notifies the Dr. Watson user-mode application. With the help of hardware support in the processor, the kernel-mode portions of NT keep user-mode applications constrained to acceptable activity and prevent user-mode applications from corrupting other applications or crossing the boundary between user mode and kernel mode other than through the gateway.



## Kernel-Mode Rules

Thus far, my explanation implies that kernel-mode device drivers and subsystems do not execute in a sandbox and can do anything they want. Well, this implication is almost true. Portions of the memory map are undefined, and consequently, those portions are invalid regardless of what tries to access them. For example, if the space between 3GB and 4GB in the address map is not defined, a device driver accessing that portion of the map will cause a processor exception. In this example, the Virtual Memory Manager will recognize that a kernel-mode device driver has tried to touch invalid memory. For exceptions that originate in user mode, a kernel-mode subsystem handles the exception.

Kernel-mode components also have rigid rules about what they can do when the processor is in different states. I'll summarize the key ideas behind these rules in the next few paragraphs, but for details, see my November 1997 column, "Inside NT's Interrupt Handling."

Each processor in an NT system has an associated Interrupt Request Level (IRQL) that changes as the processor's interrupt controller fields various software and hardware interrupts. Although IRQLs have almost nothing to do with scheduling priorities, you can think of IRQLs as priorities in the sense that the interrupt controller blocks out interrupt requests with lower IRQLs while the processor is handling interrupts with higher IRQLs. In its design, NT attempts to keep the IRQL at Passive Level,

# Inside the Blue Screen

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

where no interrupts are blocked out, as much as possible. The NT scheduler executes at Dispatch Level, and NT services hardware interrupts at even higher IRQLs.

Only when the IRQL is below Dispatch Level can kernel-mode components access pageable memory or cause scheduling operations. Pageable memory includes all user-mode application memory and portions of kernel-mode memory. Pageable memory gets its name from the fact that its data can be temporarily moved from the processor's physical memory to a paging file on disk and brought back when needed. When a kernel-mode component (such as a device driver) accesses part of the memory map referring to pageable memory that has data in a paging file, it triggers a processor exception (the same one that's triggered when a component references invalid memory), and the Virtual Memory Manager must retrieve the data. However, if the IRQL is Dispatch Level or higher, the Virtual Memory Manager cannot be invoked.

The scheduler's IRQL is Dispatch Level, so a device driver cannot yield control of a processor to another program or kernel-mode component if the IRQL is at Dispatch Level or higher. To do so would force the invocation of the scheduler, which would detect that it had been called at an illegal processor state.

## Where Do Blue Screens Come From?

So where am I going with all this information? As I stated earlier, illegal processor exceptions that user-mode applications cause usually result in application termination and a Dr. Watson message, but the rest of the system continues.

When a kernel-mode device driver or subsystem causes an illegal exception, NT faces a difficult dilemma. It has detected that a part of the operating system with the ability to access any hardware device and any valid memory has done something it wasn't supposed to do.

NT could just ignore the exception and let the device driver or subsystem continue as if nothing had happened. The possibility exists that the error was isolated and that the component will somehow recover, letting NT limp along. What's more likely is that the detected exception resulted from deeper problems--for example, from a general corruption of memory or from a hardware device that's not functioning properly. Permitting the system to continue operating will probably result in more exceptions, and data stored on disk or other peripherals can become corrupt--a risk that's too high to take.

A device driver or subsystem also might realize that something is not quite right. For example, a subsystem might call a function in a device driver when the processor IRQL is Passive Level. If the function returns and the IRQL has changed, the device driver has somehow modified the IRQL without restoring it, which reveals a bug in the driver. As device drivers and subsystems execute, they require certain operations to succeed or return results within a valid range. For instance, if the Configuration Manager tries to read a Registry file from the disk and encounters an error, the Configuration Manager might not be able to continue processing without risking damage to the Registry.

To stop a system in the face of kernel-mode exceptions and to provide a systems administrator or developer information about what has happened, NT exports the KeBugCheck function for use by kernel-mode device drivers, subsystems, and the Microkernel. This function takes a Stop Code and four more parameters that are interpreted on a per-Stop Code basis. After KeBugCheck masks out all interrupts on all processors of the system, it switches the display into blue screen mode (80 columns

# Inside the Blue Screen

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

by 50 lines text mode), paints a blue background, and begins to print information about the system's state.

## Mapping the Blue Screen

The blue screen contains five areas of text from top to bottom: the Stop Code, system information, a list of loaded drivers, the stack trace, and an administrative message. In Screen 1, blank lines separate these areas. Some areas might be missing in a blue screen if the system state is too corrupt for NT to fill them in.

```
*** STOP: 0x000000A (0x00000000,0x00000002,0x00000000,8038c240)
IRQL_NOT_LESS_OR_EQUAL*** Address 8038c240 has base at 8038c000 - Ntfs.SYS

CPUID:Genuine Intel 6.3.3 irq:1f SYSVER 0xf0000565

Dll Base DateStmp - Name
80100000 336546bf - ntoskrnl.exe
80000100 334d3a53 - atapi.sys
802aa000 33013e6b - epst.mpd
802b9000 336015af - CLASS2.SYS
802bd000 33d844be - Siwvid.sys
f9318000 31ec6c8d - Floppy.SYS
f9468000 31ed868b - KSecDD.SYS
f9358000 335bc82a - i8042prt.sys
f947c000 31ec6c94 - kbdclass.sys
f9370000 33248011 - VIDEOPORT.SYS
f9490000 31ec6c6d - vga.sys
f90f0000 332480d0 - Npfs.SYS
a0000000 335157ac - win32k.sys
fe0c9000 335bd30e - Fastfat.SYS
fe108000 31ec6c9b - Parallel.SYS
f9050000 332480ab - Serial.SYS

Dll Base DateStmp - Name
80010000 33247f88 - hal.dll
80007000 33248043 - SCSIPORT.SYS
802b5000 336016a2 - Disk.sys
8038c000 3356d637 - Ntfs.sys
803e4000 33d84553 - NTIce.sys
f95c9000 31ec6c99 - Null.SYS
f95ca000 335e60cf - Beep.SYS
f9474000 3324806f - mouclass.sys
f95cb000 3373c39d - ctrl2cap.SYS
fe9d7000 3370e7b9 - ati.sys
f93b0000 332480dd - Msfs.SYS
fe957000 3356da41 - NDIS.SYS
fe914000 334ea144 - ati.dll
fe110000 31ec7c9b - Parport.SYS
f95b4000 31ec6c9d - ParVdm.SYS

Address dword dump Build [1314] - Name
801afc24 80149905 80149905 ff8e6b8c 80129c2c ff8e6b94 8025c000 - Ntfs.SYS
801afc2c 80129c2c 80129c2c ff8e6b94 00000000 ff8e6b94 80100000 - ntoskrnl.exe
801afc34 801240f2 80124f02 ff8e6df4 ff8e6f60 ff8e6c58 80100000 - ntoskrnl.exe
801afc54 80124f16 80124f16 ff8e6f60 ff8e6c3c 8015ac7e 80100000 - ntoskrnl.exe
801afc64 8015ac7e 8015ac7e ff8e6df4 ff8e6f60 ff8e6c58 80100000 - ntoskrnl.exe
801afc70 80129bda 80129bda 00000000 80088000 80106fc0 80100000 - ntoskrnl.exe

Restart and set the recovery options in the system control panel
or the /CRASHDEBUG system start option. If this message reappears,
contact your system administrator or technical support group.
```

The administrative message tells you to contact your systems administrator if you have a chronic blue screen problem on your system. The most useful portion of the display is usually the Stop Code area. This area lists the Stop Code and the four additional parameters passed to KeBugCheck. In Screen 1, the Stop Code is 0x000000A, and the additional parameters appear inside the parentheses after the Stop Code.

The Stop Code is a number that represents the nature of the detected problem. The bugcodes.h file in the Windows NT Device Driver Kit contains a complete list of the 150 or so Stop Codes. However, you will typically encounter only 4 or 5 of them. The text line below the Stop Code provides the text equivalent of the Stop Code numeric identifier. I'll discuss some of the common Stop Codes a little later.

# Inside the Blue Screen

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

Interpreting the additional Stop Code parameters rarely provides any insight into a problem for anybody other than a device driver writer (or a member of the Microsoft NT development team). Fortunately, NT does some interpretation for us. KeBugCheck scans the parameters for one that looks like it might be an address pointing to the memory image of an Executive subsystem or a device driver. When KeBugCheck finds one, it prints the parameter, the base address of the module the parameter is in, and the name of the module. This last piece of information is crucial, and I'll describe how you can use it a little later.

The system information area of the screen is below the Stop Code area, and it simply identifies the system's processor type (e.g., Pentium, x486) and NT's base build number (no Service Pack information appears). In Screen 1, the Build Number is 0xf0000565 (1381 in decimal), which is what you'll see for any NT 4.0 installation. An IRQL number also appears in this area, but a bug in KeBugCheck causes it to record the IRQL incorrectly.

Below the system information on the blue screen is the loaded driver area. Here you'll see a listing of all the registered device drivers at the time of the stop. KeBugCheck prints the name, base memory address, and date-stamp (the time a driver was built). Unless you develop device drivers, this information is useless.

Finally, just below the loaded driver area is a snapshot of the system stack at the time of the call to KeBugCheck. Each module (except the first one) in the list had invoked the module printed on the line above it and was waiting for a result. The system detected a problem while the module on the first line was executing, and often this module matches the module shown in the Stop Code area (Ntfs.SYS in Screen 1).

## Interpreting the Blue Screen Information

So, what do you do with the data the blue screen provides? Many times, all you can do is reset the system and hope that the blue screen doesn't happen again. But sometimes an important clue is lurking in the Stop Code area or stack trace that can help you take a more proactive approach to ridding the system of the blue screen.

First, the Stop Code can provide all the information you need to identify the problem. The sidebar, "Common Stop Codes," page 62, lists several Stop Codes, their causes, and some suggestions about what to do if you encounter one. Microsoft Windows NT Workstation Resource Kit contains more information about Stop Codes.

Often, you begin seeing blue screens after you install a new software product or piece of hardware. If you've just added a driver, rebooted, and got a blue screen early in system initialization, you can reset the machine and press the space bar when instructed, to get the Last Known Good configuration. Enabling Last Known Good causes NT to revert to a copy of the Registry's device driver registration key (HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services) from the last successful boot (before you installed the driver).

If you keep getting blue screens, an obvious approach is to uninstall the things you added just before the appearance of the first blue screen. If some time has passed since you added something new or you added several things at about the same time, you need to note the names of the modules you see in both the Stop Code and stack trace areas. Note that ntoskrnl.exe refers to the image that contains all NT's core kernel-mode subsystems as well as the Microkernel.

# Inside the Blue Screen

Mark Russinovich

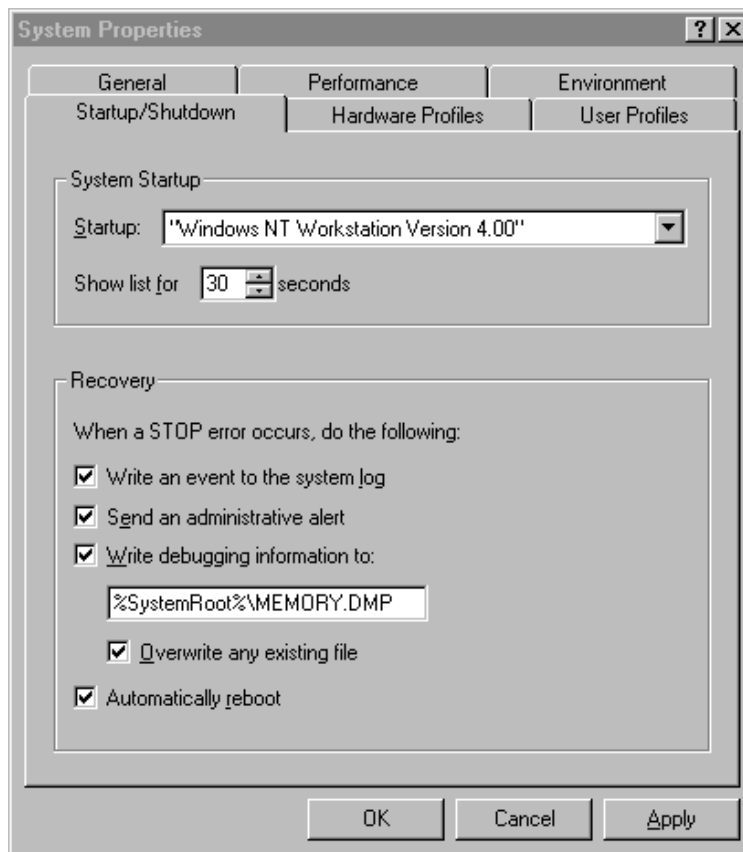
(Reprinted from WindowsItPro Magazine)

If you recognize any of the module names as being related to something you just added (such as scsiport.sys if you put on a new drive), you've possibly found your culprit. Many device drivers have cryptic names, so one thing you can do to figure out which application or hardware device is associated with a name is to run the Regedit Registry viewing tool the next time you boot the system or on a similarly equipped machine. Search for the name of the driver under the HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services key. This branch of the Registry is where NT stores registration information for every device driver in the system. If you find a match, look for a value called DisplayName. Some drivers fill in this value with a name descriptive of the device driver's purpose. For example, you might find Virus Scanner, which can implicate the antivirus software you have running.

You can also search Microsoft's online Knowledge Base (<http://www.microsoft.com>) for the Stop Code and the name of the suspect hardware or application. You might find information about a workaround, an update, or a Service Pack that fixes the problem you're having.

## Setting the Blue Screen Options

Instead of just halting the system with a blue screen, you can have NT log an event to the system log, send you an administrative alert, write a dump of the machine's physical memory to disk, or automatically reboot the computer. You can configure these options on the Startup/Shutdown tab of the System applet in Control Panel, as shown in Screen 3, page 64.



If you want to track how often a computer runs into problems, select the option to record the event in the system log, which you can view with the Event Viewer administrative tool. In general, you won't want the machine's memory written to disk unless you have a chronic problem that a particular

# Inside the Blue Screen

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

hardware vendor or Microsoft will help you debug. In this case, be prepared to copy a file as large as the computer's memory (i.e., 128MB for a 128MB machine) to send for debugging. Contact the hardware vendor or Microsoft for instructions about where and on what medium to send the dump.

Finally, automatic rebooting is an option you want to enable if your machine is performing a task for which you want to minimize downtime. If you have a Web server that configures itself automatically when NT starts, automatic rebooting after a stop will keep your site offline for as little time as possible.

## At Wits End

Unfortunately, you can't run a magical program to identify the exact cause of blue screens or make them go away. Even with extensive knowledge of NT internals and device drivers, you'll still find that reading a blue screen and trying to figure out what happened is a little like fumbling around in a dark room. However, the next time you're unpleasantly surprised with a blue display, you might find some solace knowing what's going on behind the scenes--that a subsystem or driver made a call to KeBugCheck to provide the information in the different areas of the screen.

## Common STOP Codes

Windows NT employs about 150 Stop Codes. However, you encounter the following Stop Codes most frequently. For a complete list of NT's Stop Codes, see the bugcodes.h file in the Windows NT Device Driver Kit.

### **IRQL\_NOT\_LESS\_OR\_EQUAL 0x0A**

This code is probably the most frequently appearing code, and it usually results from a buggy driver. The most common source of the problem is that the Virtual Memory Manager has detected a kernel-mode component's attempt to access pageable memory when the IRQL is Dispatch Level or higher and the memory is in the paging file. The IRQL must be below Dispatch Level for this operation to be legal. Look at the modules listed in the Stop Code and stack trace areas of the screen for a possible candidate. This code can also be a side effect resulting from a driver not shown in either area that scribbled on memory it shouldn't have.

### **UNEXPECTED\_KERNEL\_MODE\_TRAP 0x7F and KMODE\_EXCEPTION\_NOT\_HANDLED 0x1E**

These two codes also show up frequently. In this case, the Microkernel's processor exception handler has detected that a driver or subsystem has tried to execute an illegal processor instruction, or a software instruction that NT cannot interpret. The cause can be a faulty memory module or a driver that has corrupted memory. The module information on the blue screen is usually misleading in this case, making it difficult to identify the source of the problem.

### **NO\_MORE\_IRP\_STACK\_LOCATIONS 0x35**

With this code, if you've added a new virus scanner or someone has accessed a shared volume over the network for the first time on the machine, the Server device driver can be at fault. The Server device driver constructs I/O request packets with a slot for every device driver on the path to the disk. Sometimes the number of I/O request packets the Server device driver allocates is insufficient, resulting in this Stop Code. Try increasing the HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\IrpStackSize setting to a number higher than 4 (or whatever it's set to) and see whether the problem goes away.

### **INACCESSIBLE\_BOOT\_DEVICE 0x7B**

If you see this Stop Code, NT is very early in a boot and cannot access the disk partition that boot.ini is pointing to for the location of the system files (where your \winnt directory resides). The disk containing that partition is faulty, or the data on the disk or partition has become corrupt. I

## Inside the Blue Screen

Mark Russinovich

(Reprinted from WindowsItPro Magazine)

encountered this code when I left an NT 4.0 distribution CD-ROM in my CD-ROM drive and rebooted. The computer tried to boot from the CD-ROM, and NT displayed this message when it couldn't continue. An NT repair install is worth a try, but you'll likely have to buy a new driver or reformat, reinstall, and restore backed-up data.