CHAPTER 6

Support for Running MS-DOS–Based Applications

Support for MS-DOS–based applications, device drivers, and TSRs is maintained in Windows 95. In fact, Windows 95 offers better compatibility for running MS-DOS–based applications than Windows 3.1 does, including applications that are hardware-intensive, such as games.

Like Windows 3.1, Windows 95 allows users to launch an MS-DOS command prompt as an MS-DOS VM. The functionality supported in an MS-DOS VM is the same as that available under the latest version of MS-DOS, allowing users to run the same intrinsic commands and utilities.

Windows 95 delivers great support for running MS-DOS–based applications and enables even applications that would not run under Windows 3.1 to run properly. This support allows MS-DOS–based applications to coexist peacefully with the rest of the Windows 95 environment.

Summary of Improvements over Windows 3.1

Improvements made to the system provide the following benefits for running MS-DOS–based applications in the Windows 95 environment:

- Zero conventional memory footprint for protected-mode components
- Improved compatibility for running MS-DOS-based applications
- Improved robustness for MS-DOS–based applications
- Better support for running MS-DOS–based games, including in a window
- Support for running existing MS-DOS–based applications without exiting Windows 95 or running MS-DOS externally
- Consolidated attributes for customizing the properties of MS-DOS–based applications
- The availability of the Toolbar when running an MS-DOS–based application in a window, providing quick access to features and the functionality for manipulating the window environment
- A user-scaleable MS-DOS window through the use of TrueType fonts

- The ability to gracefully end an MS-DOS–based application without exiting the application
- The ability to specify local VM environment settings on a per-application basis through the use of a separate batch file
- Support for new MS-DOS commands, providing tighter integration between the MS-DOS command line and the Windows environment

Zero Conventional Memory Footprint Components

Windows 95 helps make the maximum amount of conventional memory available for running existing MS-DOS–based applications. Some MS-DOS–based applications did not run under Windows 3.1 because by the time MS-DOS–based device drivers, MS-DOS–based TSRs, MS-DOS–based networking components, and Windows 3.1 were loaded, not enough conventional memory was available. Windows 95 replaces many of the 16-bit real-mode components with 32-bit protected-mode counterparts to provide the same functionality while improving overall system performance and using no conventional memory.

32-bit virtual device drivers are provided to replace their 16-bit real-mode counterparts for functions such as those listed in the table on the facing page.

The memory savings that results from using 32-bit protected-mode components can be quite dramatic. For example, if a PC were configured with the NetWare NetX client software and used a SCSI CD-ROM drive, SMARTDrive, the MS-DOS mouse driver, and DriveSpace disk compression, the conventional memory savings that would result from using Windows 95 would be *over* 262 KB!

Functions Carried Out by 32-Bit Device Drivers in Windows 95		
Description	**File(s)**	**Conventional Memory Saved**
Microsoft Network client software	NET.EXE (full) PROTMAN NETBEUI EXP16.DOS (MAC)	95 KB 3 KB 35 KB 8 KB
Novell NetWare client software	LSL EXP16ODI (MLID) IPXODI.COM NETBIOS.EXE NETX.EXE VLM.EXE	5 KB 9 KB 16 KB 30 KB 48 KB 47 KB
MS-DOS extended file sharing and locking support	SHARE.EXE	17 KB
Adaptec SCSI driver	ASPI4DOS.SYS	5 KB
Adaptec CD-ROM driver	ASPICD.SYS	11 KB
Microsoft CD-ROM Extensions	MSCDEX.EXE	39 KB
SmartDrive disk caching software	SMARTDRV.EXE	28 KB
Microsoft Mouse driver	MOUSE.COM	17 KB
Microsoft DriveSpace disk compression driver	DRVSPACE.BIN	37 KB

Try It!

Test Conventional Memory Savings

1. Install Windows 95 on a PC with a configuration similar to one that now runs Windows 3.1. For example, use PCs with SCSI drivers, network drivers, and system support files, such as SMARTDRV, MSCDEX, or SHARE.
2. With MS-DOS–based device drivers and TSRs loaded on both machines, type *mem /c* at a command prompt command under Windows 3.1 and under Windows 95. Notice the memory savings under Windows 95 for the same configuration.

Compatibility Improvements

For a number of reasons, some MS-DOS–based applications wouldn't run properly under Windows 3.1. For example, some MS-DOS–based applications required that lots of free conventional memory be available and thus were prevented from running in an MS-DOS VM by large real-mode components, such as network drivers or device drivers. Other MS-DOS–based applications wouldn't run under Windows 3.1 because they required direct access to the computer hardware and conflicted with Windows internal drivers or other device drivers.

The MS-DOS support goal of Windows 95 is to be able to run the "clean" MS-DOS–based applications that ran under Windows 3.1 *and* the "bad" MS-DOS–based

applications that tried to take over the hardware or required machine resources unavailable under Windows 3.1.

Many MS-DOS–based games assume that they are the only application running on the system and access and manipulate the underlying hardware directly, thus preventing them from being run in an MS-DOS VM under Windows 3.1. Games are the most notorious class of MS-DOS–based applications that don't get along well with Windows 3.1. Some of these applications write to video memory directly, manipulate hardware support resources such as clock timers, and take over hardware resources such as sound cards.

A number of strategies have been used to provide better support for running MS-DOS–based applications that interact with the hardware, including better virtualization of computer resources such as timers and sound devices. In addition, the use of 32-bit protected-mode device drivers benefits MS-DOS–based applications by providing more free conventional memory than was available under Windows 3.1, so that memory-intensive applications run properly.

Different MS-DOS–based applications require varying levels of support from both the computer hardware and from the operating system. For example, some MS-DOS–based games require close to 100 percent use of the CPU to perform properly, and other MS-DOS–based applications modify interrupt addresses and other low-level hardware settings. Windows 95 provides several different levels of support for running MS-DOS–based applications. These levels of support take into account that different applications interact with the hardware in different ways, and that some behave well, whereas others expect exclusive access to the PC system and hardware. By default, MS-DOS–based applications are preemptively multitasked with other tasks running on the system and can run either full-screen or in a window. (CPU-intensive MS-DOS–based applications may not perform well in a window but can be run in full-screen mode to get the best response level.)

APPS.INF

Windows 95 provides an INF file that contains program settings for many of the popular MS-DOS–based applications. These program settings are based on results from testing of Windows 95 and specify and special configuration-related settings that are necessary to allow the application to run under Windows 95.

The APPS.INF file is processed when the user attempts to execute an MS-DOS–based application from the Windows 95 user interface. If a program information file (PIF) doesn't exist for the MS-DOS–based application, the APPS.INF file will be examined by the system for information about the specified MS-DOS–based application. If the application is listed in the APPS.INF file, the system will read the contents and create a PIF that will be used when the application is executed.

MS-DOS Mode

To provide support for the most intrusive set of MS-DOS–based applications that work only under MS-DOS and require 100 percent access to the system components and system resources, Windows 95 provides a mechanism that is the equivalent of running an MS-DOS–based application in real-mode MS-DOS. This mechanism, called MS-DOS mode, provides an "escape hatch" for applications that run only under MS-DOS. In this mode, Windows 95 removes itself from memory (except for a small stub) and provides the MS-DOS–based application with full access to all the resources in the computer. Relatively

few MS-DOS–based applications need to run in single MS-DOS application mode because of the improved compatibility support provided by Windows 95.

To run an MS-DOS–based application in this mode, the user sets the MS-DOS Mode property on the Advanced Program Settings dialog (from the Advanced button on the Program tab) of the MS-DOS property sheet for the application. To create a unique environment tailored to an application's needs and system requirements, the user can also specify a CONFIG.SYS or AUTOEXEC.BAT file to run for the application. When the user runs an MS-DOS–based application in this mode, Windows 95 asks whether running tasks can be ended. With the user's approval, Windows 95 ends all running tasks, configures the machine to use the CONFIG.SYS and AUTOEXEC.BAT files for the MS-DOS mode session, restarts the computer, loads a real-mode copy of MS-DOS, and launches the specified application. This process is like exiting Windows 3.1 and then running the specified MS-DOS–based application under MS-DOS. When the user exits the MS-DOS–based application, Windows 95 restarts and returns the user to the Windows 95 shell. This solution is much more elegant than requiring users to dual-boot between different operating systems in order to run desired applications.

Some entries in the APPS.INF file contain setting information that instruct Windows 95 to run an MS-DOS–based application in *MS-DOS mode*. For these applications, they will only run in MS-DOS mode due to problems they have running in the protect-mode environment of Windows 95, or due to assumptions they make about the environment (e.g., addressing of extended memory for loading their information) that prevent them from running under Windows 95.

Try It!

Run an MS-DOS–Based Application

1. Identify an MS-DOS–based application that you know does not run under Windows 3.1, and run it under Windows 95. Does it work?

2. Identify an MS-DOS–based application that you know runs under Windows 3.1 full-screen but not in a window, and run it under Windows 95 in a window. Does it work?

Support for Graphic-Intensive MS-DOS–Based Applications

Windows 95 improves the support for running graphic-intensive MS-DOS–based applications in the Windows environment. MS-DOS–based applications that use VGA graphic video modes can now be run in a window; they no long have to be run in full-screen mode as with Windows 3.1. Users can still choose to run graphic-intensive MS-DOS–based applications in full-screen mode for the best level of performance.

Memory Protection

To support a higher level of memory protection for running MS-DOS–based applications, Windows 95 includes on the Program property sheet a global memory protection attribute that allows the MS-DOS system area to be protected from errant MS-DOS–based applications. When the global memory protection attribute is set, the MS-DOS system

area sections are read-protected so that applications can't write to this memory area and corrupt MS-DOS support and MS-DOS–based device drivers. In addition to system area protection, enhanced parameter validation is performed for file I/O requests issued through the MS-DOS Int 21h function, providing a higher degree of safety.

This option is not enabled by default for all MS-DOS–based applications because of the additional overhead associated with providing improved parameter and memory address checking. Users can set this flag if they constantly have difficulty running a specific MS-DOS–based application.

Better Defaults for Running MS-DOS–Based Applications

By default, Windows 3.1 ran MS-DOS–based applications full screen and disabled the ability of MS-DOS–based applications to run in the background. To change this default behavior for a specific MS-DOS–based application, users had to use the PIF Editor (PIFEDIT) application to modify or create a program information file (PIF) for the application.

By default, Windows 95 runs MS-DOS–based applications in a window and enables the background-execution setting, allowing the application to continue to run when it is not the active application. This change in default behavior provides better integration of running MS-DOS–based applications and Windows–based applications without requiring users to change or customize the state of the system.

Consolidated Customization of MS-DOS–Based Application Properties

Each MS-DOS–based application has different characteristics and mechanisms for using machine resources such as memory, video, and keyboard access. Both Windows 95 and Windows 3.1 understand how to run Windows–based applications because requests for system services are handled through the use of the Windows API. However, MS-DOS–based applications include only minimal information about their requirements in the format of their .EXE headers. To provide additional information about their requirements to the Windows environment, PIFs are used to specify the necessary configuration settings.

Under Windows 3.1, the PIF Editor application, shown in Figure 39, was used to create or change properties associated with running MS-DOS–based applications. Problems associated with the PIF Editor and the PIF creation process included difficulty in accessing the PIF Editor and PIF settings, the lack of association for new users between PIF properties and the MS-DOS–based application, the lack of a single location for storing PIF files (other than placing them all in the Windows directory), and less-than-intelligent defaults for running MS-DOS–based applications.

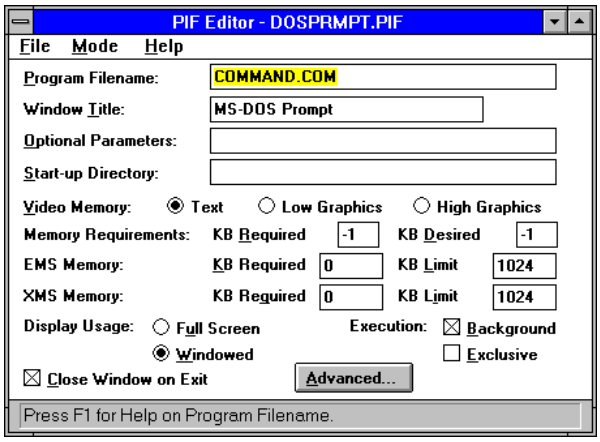

Figure 39. The PIF Editor in Windows 3.1

Windows 95 enhances the ability to define properties for running MS-DOS–based applications by consolidating PIF files into a single location (the PIF directory where Windows 95 is installed), providing easy access to property information for an application (right-clicking the application's icon or window), and providing better organization of properties (through the use of a tabbed property sheet, shown in Figure 40). By means of these improvements, Windows 95 provides greater flexibility and control for running MS-DOS–based applications.

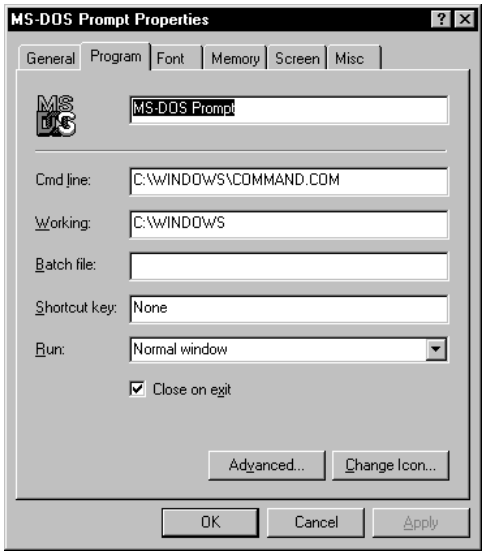

Figure 40. The property sheet for configuring an MS-DOS–based application

Try It!

View the Property Sheet for an MS-DOS–Based Application

1. Right-click the icon for an MS-DOS–based application, and choose *Properties*.
2. Right-click the title bar of an active MS-DOS–based application, and choose *Properties*.

Toolbars in MS-DOS Windows

In addition to providing compatibility enhancements to better support the running of MS-DOS–based applications, Windows 95 makes using MS-DOS–based applications in the Windows environment even easier than Windows 3.1. Many Windows–based applications provide one or more toolbars for quickly accessing common features and functionality, Windows 95 extends this simple but powerful feature to provide easy access to the functionality associated with an MS-DOS–based application, as shown in Figure 41.

Figure 41. A toolbar in a windowed MS-DOS box

Optionally, users can enable the display of a toolbar in the window of a running MS-DOS–based application to provide the user with quick access to the following functionality:

- Simpler access to cut, copy, and paste operations for integrating text or graphic MS-DOS–based applications with Windows–based applications
- Easy switching from windowed to full-screen mode
- Quick access to property sheets associated with the MS-DOS–based application
- Access to MS-DOS VM tasking properties, such as exclusive or foreground processing attributes
- Easier access to font options for displaying text in a windowed MS-DOS VM

Scaleable MS-DOS Windows

Windows 95 supports the use of a TrueType font in a windowed MS-DOS VM, which allows users to scale the MS-DOS window to any size. When the font size is set to Auto, the contents of the MS-DOS window are sized automatically to display the entire window within the user-specified area. Figure 42 shows the size of the MS-DOS command prompt window being made smaller.

Figure 42. Because of TrueType font support, the contents of this MS-DOS window will be scaled to fit the smaller size of the window so that they are still displayed in their entirety.

Try It!

Scale an MS-DOS Window

1. Open an MS-DOS VM window, click the *Font* tab on the property sheet, and check that the font size is set to *Auto*.
2. Point to the scale region in the lower-right corner of the window, change the size of the window, and notice how the window's contents are rescaled. (This functionality is more noticeable when performed at higher resolutions.)

Ending MS-DOS–Based Applications Gracefully

Windows 95 provides support for gracefully closing an MS-DOS VM through a property sheet setting that is available on an application-by-application basis. When this setting is enabled, users can close the MS-DOS–based application just as they would a Windows–based application—by clicking the Close Window button.

In addition to gracefully ending an MS-DOS–based application, robustness improvements made to the Windows 95 system ensure that system clean-up is completed properly and that all allocated resources are freed. As a result, memory used by an MS-DOS–based application running under Windows 95 is deallocated properly and made available for use by other applications (unlike under Windows 3.1, which didn't properly free DPMI memory).

Local Virtual Machine Environment Settings

When Windows 3.1 started, it used the MS-DOS environment specified before it started as the default state for each subsequently created MS-DOS VM. Any TSRs or other memory resident software that was loaded before Windows started was replicated across all MS-DOS VMs, whether the VM needed it or not. With Windows 3.1, users could not

run a batch file to set the VM environment before starting a given MS-DOS–based application. (Actually, users could run a batch file under Windows 3.1, but when the batch file finished processing the command statements, the MS-DOS VM was closed.)

Under Windows 95, a batch file can be optionally specified for a given MS-DOS–based application, allowing customization of the VM on a local basis before running the MS-DOS–based application. The batch file is specified on the Program tab of the MS-DOS–based application's property sheet, as shown in Figure 43. Using a batch file allows MS-DOS environment variables to be set or customized for individual MS-DOS–based applications and for TSRs to be loaded in the local VM only. This mechanism is like having different AUTOEXEC.BAT files for different MS-DOS–based applications.

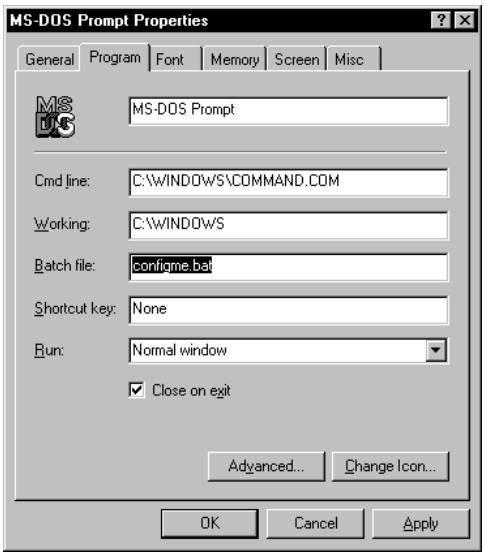

Figure 43. Specifying a batch file on the Program tab of the MS-DOS–based application property sheet

Support for Accessing Network Resources with UNC Pathnames

Windows 95 makes accessing network resources from the MS-DOS command prompt easier by supporting the use of universal naming conventions (UNC). UNC provides a standard naming scheme for referencing network servers and shared directories. It uses the following syntax:

*servername**sharename*[*pathname*]

The Windows 95 shell allows users to browse and connect to network servers without mapping a drive letter to the network resource. Windows 95 supports the same functionality at an MS-DOS command prompt and allows the user to do the following:

- View the contents of shared directories on network servers from both Microsoft Network servers and Novell NetWare servers by typing *dir* *servername**sharename*[*pathname*]

- Copy files from the contents of shared directories on network servers from both Microsoft Network servers and Novell NetWare servers by typing *copy \\servername\sharename\pathname\file destination*
- Run applications from shared directories on network servers for both Microsoft Network servers and Novell NetWare servers by typing *\\servername\sharename\pathname\filename*

New MS-DOS Prompt Commands

The MS-DOS command processor and utilities have been enhanced to provide better integration between MS-DOS functionality and the Windows environment. Commands that manipulate files have been extended to support long filenames, and some new commands have been added to Windows 95 to provide access to new capabilities supported by the system.

For example, the **start** command, which has the following syntax:

start <*application name*> | <*document name*>

allows users to start an MS-DOS or Windows–based application from the command prompt in one of the following ways:

- Start an application by specifying the name of a document to open, and Windows 95 launches the application associated with the given file type. For example, typing *start myfile.xls* starts the application associated with the specified file, if there is a valid association.
- Start an MS-DOS–based application in a different MS-DOS VM instead of the current one.
- Start a Windows–based application from an MS-DOS command prompt. Typing the name of a Windows–based application is essentially the same as typing *start <application>*.

Try It!

Launch an Application from the MS-DOS Command Prompt

1. Type *start /?* to see the options available.
2. Type *start edit* to start the MS-DOS Edit application in another VM.
3. Type *start /m notepad* to start the Notepad Windows–based application in minimized form.

Support for Long Filenames

Many MS-DOS intrinsic commands and utilities have been extended to support the use of long filenames. For example, the following commands are among those that have been extended to support long filenames:

- The **dir** command has been extended to show long filenames in the directory structure, along with the corresponding 8.3 filename. Also, the **dir** command now supports a verbose mode so that users can display additional file details by typing *dir /v*.
- The **copy** command has been extended to allow mixing of long and short filenames in copy operations. For example, typing *copy myfile.txt "this is my file"* creates a new file with a long filename.