CHAPTER 5

Improved Support for Running MS-DOS–based Applications

Support for MS-DOS–based applications, device drivers, and terminate-and-stay-resident programs (TSRs) does not go away 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.

As with Windows 3.1, Windows 95 provides the ability for a user to launch an MS-DOS command prompt as an MS-DOS virtual machine (VM). The functionality supported in an MS-DOS VM, is the same functionality that is 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, allowing even applications that would not run under Windows 3.1 to run properly. This 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 in 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 properties of MS-DOS–based applications
- Toolbar availability when running an MS-DOS–based application in a window providing quick access to features and functionality to manipulate the window environment
- User-scaleable MS-DOS window through the use of TrueType fonts
- Ability to gracefully end MS-DOS–based application without exiting the application
- 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 Footprint Components

Windows 95 helps to provide the maximum amount of conventional memory available for running existing MS-DOS–based applications. Some MS-DOS–based applications do 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, there was not enough conventional memory available. Windows 95 provides 32-bit protected-mode components that replace many of the 16-bit real-mode counterparts, providing the same functionality while improving overall system performance and using no conventional memory.

32-Bit virtual device drivers are provided to replace the 16-bit real-mode counterparts for such functions as:

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

The resulting memory savings for using 32-bit protected-mode components can be quite dramatic. For example, suppose a PC was configured with the NetWare NetX client software, using a SCSI CD-ROM drive, and with the MS-DOS support files SMARTDrive, Mouse driver, and DriveSpace disk compression. The resulting conventional memory *savings* that Windows 95 would offer this configuration would be *over* 262 KB!

Improved Compatibility

There are a number of reasons why some MS-DOS–based applications do not run properly under Windows 3.1. For example, some MS-DOS applications required lots of available free conventional memory, and thus wouldn't run in a DOS VM due to large real-mode components, such as network drivers or device drivers. Other MS-DOS–based applications would not run under Windows 3.1 because they required direct access to the computer hardware and conflicted with Windows internals or other device drivers.

The goal of Windows 95 to support running MS-DOS–based applications is to be able to run existing "clean" MS-DOS–based applications that ran under Windows 3.1, as well as to support running 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 take advantage of the assumption that they are the only application running in the system, and access and manipulate the underlying hardware directly, thus preventing them from being run in a 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 the hardware support resources such as clock timers, and take over hardware resources such as sound cards.

A number of things have been done 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 device support. In addition, the use of 32-bit protected-mode device drivers benefits MS-DOS–based applications by providing them with more free conventional memory than was available under Windows 3.1, allowing a class of memory-intensive applications to run properly.

Different MS-DOS–based applications require varying levels of support from both the computer hardware and from the operating system. For example, there are some MS-DOS–based games that require close to 100% use of the CPU to perform properly, and there are other MS-DOS–based applications that 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—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 run well in a window for performance reasons, but can be run in full-screen mode to get the best response level.)

Single MS-DOS Application Mode

To provide support for the most intrusive set of MS-DOS–based applications that only work under MS-DOS and require 100% access to the system components and system resources, Windows 95 provides a mechanism that is the equivalent of running an MS-DOS–based application from real-mode MS-DOS—this mechanism is called Single MS-DOS application mode. While fewer MS-DOS–based applications will need to run in this mode due to improved compatibility support provided by Windows 95, this mode provides an "escape hatch" mechanism for running applications that only run under MS-DOS. Users can also specify the name of a CONFIG.SYS or AUTOEXEC.BAT file to run for the specified application to create a unique environment custom tailored for that application's needs and system requirements.

To run an MS-DOS–based application in this mode, users set the Single MS-DOS Application Mode property from the Program tab on the MS-DOS property sheet for the application. 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. Before a user runs an MS-DOS–based application in this mode, Windows 95 prompts the user as to whether running tasks can be ended. Upon user's approval, Windows 95 ends all running tasks, loads a real-mode copy of MS-DOS, and launch the specified application. This process is like exiting Windows 3.1, then running the specified MS-DOS–based application under MS-DOS. Once 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 the user to dual-boot between different operating systems in order to run their desired applications. Windows 95 makes it easy for users to run their existing applications from within the Windows 95 environment.

Improved Support for Graphic-intensive MS-DOS–based Applications

Windows 95 improves the support for running MS-DOS–based applications in the Windows environment by providing better support for running graphic-based applications in a window, rather than requiring the application to be run in full-screen mode as with Windows 3.1. MS-DOS–based applications that use VGA graphic video modes can now be run in an MS-DOS window, whereas under Windows 3.1 the user was prevented from doing this. While Windows 95 is improved over Windows 3.1, the user may choose to run graphic-intensive MS-DOS–based applications in full-screen mode for the best level of performance.

Improved Memory Protection

To support a higher level of memory protection for running MS-DOS–based applications, Windows 95 includes a "global memory protection" attribute on the Program property sheet tab 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 into this memory area and corrupt MS-DOS support and MS-DOS–based device drivers. In addition to the 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 due to the additional overhead associated with providing improved parameter and memory address checking. Users would set this flag if they are constantly encountering difficulty running a specific MS-DOS–based application.

Better Defaults for Running MS-DOS–based Applications

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

Windows 95 defaults to running 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. The change in this default behavior provides better integration between running MS-DOS–based applications and Windows–based applications without requiring the user 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. Windows 95 (and Windows 3.1) understand how to run Windows–based applications as requests for system services is handled through the use of the Windows API. However, MS-DOS–based applications only included minimal information about their requirements in the format of the .EXE header associated with each application. To provide additional information to the Windows environment about the requirements for running MS-DOS–based applications, a program information file (.PIF) is used to specify the configuration settings used to run MS-DOS–based applications in the Windows environment.

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

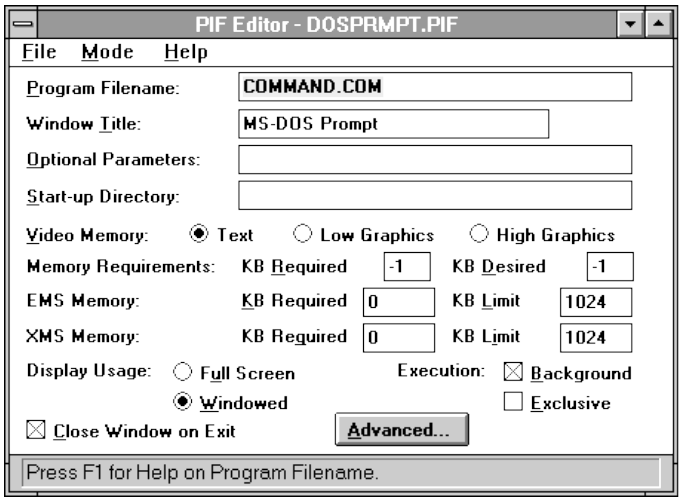

Figure 1. PIF Editor in Windows 3.1

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 (using the right mouse button to simply click the icon or application window), and simplifying the user interface to provide better organization of property settings (through the use of a tabbed property sheet dialog box). Through the use of property sheets, Windows 95 provides greater flexibility and control for running MS-DOS–based applications.

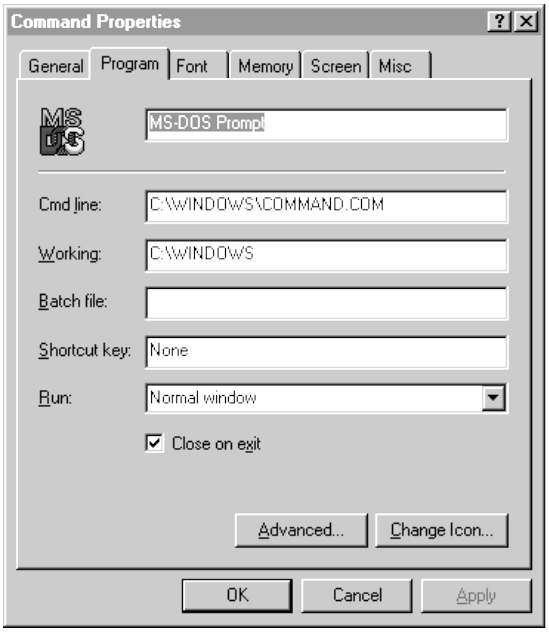

Figure 2. Property Sheet for Configuring an MS-DOS–Based Application

Toolbar in MS-DOS Window

In addition to providing compatibility enhancements in Windows 95 to support running MS-DOS–based applications better than under Windows 3.1, Windows 95 makes it even easier to use MS-DOS–based applications in the Windows environment than Windows 3.1. Many Windows–based applications implement a toolbar to provide quick access to common features and functionality of a product, Windows 95 extends this simplicity and power feature to making it easy to access functionality associated with an MS-DOS–based application.

Figure 3. Toolbar in Windowed MS-DOS Box

Optionally, a user 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-based or graphics-based MS-DOS–based applications with Windows–based applications
- Easy access to switching from windowed to full-screen mode
- Quick access to property sheet information 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 use in displaying text in a windowed MS-DOS VM

User-Scalable MS-DOS Window

Windows 95 supports the use of a TrueType font in a windowed MS-DOS VM, supporting the ability for a user to scale the MS-DOS window to any size. When the font size is set to "Auto," the MS-DOS window is sized automatically to display the entire window within the user-specified area. Figure 4 shows the MS-DOS command prompt window being changed to a smaller size.

Figure 4. With TrueType Font Support, Users Can Scale an MS-DOS Window

Ending MS-DOS–based Applications Graceful

Windows 95 provides support for gracefully closing an MS-DOS VM through a property sheet setting available on an application-by-application basis. When enabled, the user can close an MS-DOS–based application just as a Windows–based application is closed—by clicking the close window button.

In addition to simply ending an MS-DOS–based application, robustness improvements made to the Windows 95 system ensure that system clean up is completed properly and all allocated resources are freed. This results in memory used by the MS-DOS–based applications is deallocated properly and available for use by other applications. (Windows 3.1 didn't properly free DPMI memory, for example.)

Local Virtual Machine Environment Settings

When Windows 3.1 starts up, it uses the MS-DOS environment as specified before Windows is started as the default state for each MS-DOS VM that is created. Any TSRs or other memory resident software that is loaded before starting Windows is replicated across all MS-DOS VMs, whether the VM needs it or not. Windows 3.1 did not provide a mechanism to allow a user to run a batch file that set the VM environment, before starting a given MS-DOS–based application. Actually, a batch file could be run by the user under Windows 3.1, but once 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. This 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 is like having a separate AUTOEXEC.BAT for different MS-DOS–based applications. The batch file is specified on the Program tab of the property sheet for the MS-DOS–based application.

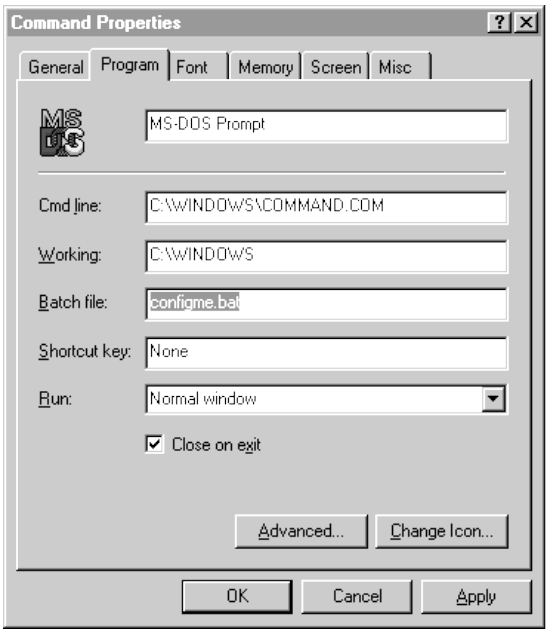

Figure 5. Property Sheet Tab for Specifying Program Attributes

Support for UNC Pathnames to Access Network Resources

Windows 95 makes it even easier to access network resources from the MS-DOS command prompt by supporting the use of universal naming conventions (UNC). UNC names provide a standard naming scheme to reference network servers, and shared directories and use 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:

- View the contents of shared directories on network servers from both Microsoft Network servers and Novell NetWare servers: **dir** \
  *servername**sharename*[*pathname*]
- Copy files from the contents of shared directories on network servers from both Microsoft Network servers and Novell NetWare servers: **copy** \
  *servername**sharename**pathname**file destination*
- Run applications from shared directories on network servers for both Microsoft Network servers and Novell NetWare servers: \
  *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 file names, and some new commands have been added to Windows 95, providing access to new capabilities supported by the system.

Starting MS-DOS *and* Windows–based Applications

For example, the **start** command allows a user to start a Windows–based or MS-DOS–based application from the command prompt in one of the following ways:

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

- Start an application by specifying the name of a document to open, and Windows 95 will launch the application associated with the given file type. For example, a user can type "**start myfile.xls**" and the application associated with the file specification will start, 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. When the user just types the name of a Windows–based application it is essentially the same as typing "**start** <*application*>."

Support for Long File Names

Many MS-DOS intrinsic commands and utilities have been extended to support the use of long file names. Basic examples of support for long file names includes the following commands. Many other commands have also been extended.

- The **dir** command has been extended to show long file names in the directory structure, along with the corresponding 8.3 filename. Also, the **dir** command now supports a "verbose" mode to display additional file details by typing "**dir /v**".
- The **copy** command has been extended to allow copying or long file names to/from short or long file names. For example, typing: "**copy myfile.txt "this is my file"**" will create a new file with a long file name.

Try It!

To see how Windows 95 improves support for running MS-DOS–based applications over Windows 3.1, you've got to try it!

Improved Support for Running MS-DOS–based Applications

- Try an MS-DOS–based application that is known to not run under Windows 3.1 and run it under Windows 95. Does it work? (If not, report it as a bug)
- Take an MS-DOS–based application that is known to run under Windows 3.1, but doesn't run in a window, and run it under Windows 95 in a window. Does it work? (If not, report it as a bug)

More Free Conventional Memory

Install Windows 95 on a PC with a configuration similar to one now running Windows 3.1 with MS-DOS–based device drivers and TSRs loaded. For example, use PCs with SCSI drivers, network drivers, or system support files such as SMARTDRV, MSCDEX, or SHARE.

- Type the "**mem /c**" command under Windows 3.1 and under Windows 95. Is there a memory savings under Windows 95 for the same configuration?

MS-DOS–based Application Property Sheets

To see a property sheet for an MS-DOS–based application, try the following:

- Use the right mouse button to click the icon for an MS-DOS–based application, and then click Properties.
- Use the right mouse button to click the title bar of an active MS-DOS–based application, and then click Properties.

Scalable MS-DOS Window

To demonstrate the ability to scale an MS-DOS window, open an MS-DOS VM window and be sure the font size is set to "Auto" from the Font tab on the property sheet.

Click the mouse in the scale region of the lower-right corner of the window and change the size—this functionality is more noticeable when performed at higher resolutions.

Launching Applications from the MS-DOS Command Prompt

To demonstrate the ability for launching applications under Windows 95 from the MS-DOS command prompt, try the **start** command in a variety of scenarios. From an MS-DOS command prompt, try these operations:

- Type "**start /?**" to see the options available.
- Type "**start edit**" to start the MS-DOS Edit application in another VM.
- Type "**start /(your app)**" to try running one of your own MS-DOS–based appplications.

BLANK PAGE

IMPORTANT: This text will appear on screen, but will not print on a PostScript printer.

This page should be the last one in this file; it was inserted by running the InsertBlankPage macro.

Do not type any additional text on this page!