Installation and Update Guide

Microsoft® WINDOWS™ DEVICE DRIVER KIT

Microsoft® Windows™

Version 3.1

Installation and Update Guide

For the Microsoft Windows Operating System

Microsoft Corporation

Document No. PC29131-0392

Contents

Introduction

About the Device Driver Kit

The Microsoft® Windows™ Device Driver Kit (DDK) provides documentation, tools, and sample sources that you need to develop Microsoft Windows device drivers and virtual devices.

Windows device drivers provide the device-specific Windows support to provide Windows-based applications access to displays, printers, keyboards, pointing devices, and other input and output devices.

Windows virtual devices provide the device management Windows needs when operating in 386 enhanced mode. Virtual devices ensure that the computer's actual devices continue to operate correctly, even though two or more applications access the devices at the same time.

The DDK provides what you need to build, test, and debug new Windows device drivers and virtual devices. It also provides what you need to modify (if necessary) existing drivers and virtual devices to ensure that they work properly under the current version of Windows.

Who Uses the Device Driver Kit?

Use the DDK if you have a hardware device that is not 100 percent compatible with the devices supported by the Windows 3.1 retail device drivers, or if you want to offer Windows users access to unique features of your device.

Independent hardware vendors (IHV) use the DDK to create device drivers and virtual devices for new hardware devices. Independent software vendors (ISV) use the DDK to customize existing drivers and virtual devices so that their applications can take better advantage of the device.

Contents of the Device Driver Kit

The DDK contains sample sources, books, online information files, and tools for developing drivers and virtual devices for the following:

- Standard-mode Windows
- 386 enhanced-mode Windows
- Microsoft Windows for Pen Computing
- Microsoft Windows Multimedia

Additionally the DDK contains test suites to verify the performance of display, printer, and network drivers and qualify them for participation in the Windows 3.1 Driver Library Program.

The DDK is provided on both CD-ROM and 3.5-inch, 1.44-megabyte disks. (5.25-inch, 1.2-megabyte disks are available by calling Microsoft at 800–426–9400.) You install the DDK by copying the disk contents to your hard drive. For more information about installing the DDK, see Chapter 1, "Getting Started."

Sample Sources

The sample sources form the core of the DDK. The samples provide clear illustrations of how device drivers and virtual devices should be structured as well as how functions and services should be implemented. Whenever possible, you should use the samples sources as the starting point for the development of your own device driver or virtual device.

The DDK includes the following sample device drivers.

Driver	Devices
Communications devices	ISA-compatible serial and parallel devices.
Displays (including fonts)	CGA, EGA, VGA, Super VGA, 8514/a, and Video Seven.
Keyboards	IBM PC/AT-style keyboard.
Minidrivers (used with the Universal Printer Driver)	HP LaserJet Series II, Epson 9-pin, Epson 24-pin, IBM Proprinter 24-pin, HP PaintJet.
Multimedia	Ad Lib FM Synthesizer, IBM Game Adapter or Joystick, Pioneer 4200 Videodisc Player, Sound Blaster.
Networks	Microsoft Network (MS®-NET).

Driver	Devices
Pointing devices	Microsoft-compatible mouse.
Printers	PCL/HP LaserJet, Adobe PostScript®, and Generic/ Text Only.
Sound	System speakers.

The DDK includes the sample virtual devices for displays, pointing devices, keyboards, disks, serial ports, DMA devices, parallel ports, network drivers, NetBIOS, Extended BIOS, and ROM BIOS. For a complete list of the drivers, see DRIVERS.TXT on the Device Drivers Samples and Tools Disk #1.

Manuals in the Device Driver Kit

The DDK provides comprehensive information about Windows device drivers and virtual devices. Take a few moments to review the following descriptions of the DDK manuals to help you to determine what path to take through the information.

Microsoft Windows Installation and Update Guide describes how to install the DDK, and build Windows device drivers and virtual devices. It also provides information about what is new or updated for Windows version 3.1.

Microsoft Windows Device Driver Adaptation Guide describes how to write drivers for displays, printers, and other devices.

Microsoft Windows Minidriver Development Guide provides a task-oriented approach to using Microsoft UniTool to build a printer driver that uses the Windows Universal Printer Driver (new for Windows 3.1).

Microsoft Windows Multimedia Device Adaptation Guide describes the development of the audio, Media Control Interface (MCI), and joystick device drivers.

Microsoft Windows Printers and Fonts Kit provides information on the Printer Font and Cartridge Metrics (PFM and PCM) file formats and the PFM Editor. This manual includes technical notes on the PCL/HP LaserJet and Adobe PostScript printer drivers.

Microsoft Windows Virtual Device Adaptation Guide describes how to write virtual devices for devices such as the display, keyboard, and hard and floppy drives.

Online Information

Much of the information in the DDK books is also available online in either Microsoft QuickHelp (QH.EXE) or Microsoft Windows Help (WINHELP.EXE) formats. The DDK includes the following online information files.

File	Description
DDAG31QH.HLP	*Device Driver Adaptation Guide* in QuickHelp format.
DDAG31WH.HLP	*Device Driver Adaptation Guide* in Windows Help format.
PFK31QH.HLP	*Printers and Fonts Kit* in QuickHelp format.
PFK31WH.HLP	*Printers and Fonts Kit* in Windows Help format.
VDAG31QH.HLP	*Virtual Device Adaptation Guide* in QuickHelp format.
VDAG31WH.HLP	*Virtual Device Adaptation Guide* in Windows Help format.
W31MDQH.HLP	*Multimedia Device Adaptation Guide* in QuickHelp format.
W31MDWH.HLP	*Multimedia Device Adaptation Guide* in Windows Help format.

Technical Development Support

The following list outlines development support for the DDK.

- CompuServe Information Service (CIS).

  Microsoft sponsors "The Microsoft Connection" area (GO MICROSOFT) on CompuServe, which consists of public forums and libraries in three categories: end users, developers, and vendors of Windows-based software. These forums provide self-help and technical assistance for most Microsoft products.

  The forum for Windows Device Driver (and Software Development) Kit support provides peer-to-peer support for device driver questions, and its public libraries include a wealth of information such as sample programs, utilities, drivers, files, and development tools.

- Microsoft Knowledge Base and Software Library.

  Microsoft also offers development Q&A and applications information through the Microsoft Knowledge Base, a product database that contains information on verified product bugs (by version), workarounds, and documentation errors.

  The Microsoft Software Library offers a wide variety of product files with coding examples and demonstrations. A free product suggestion and problem report form is available so that customers can easily submit suggestions or problems directly to Microsoft.

Note For more information about signing up for CompuServe and receiving a free $15 usage credit, call (800)848–8199 and ask for operation #230. Or call Microsoft Developer Services at (800)227–4679 for additional support options.

Windows Driver Library

The Windows 3.1 Driver Library (WDL) is a growing collection of device drivers and virtual devices from which Windows users may copy the latest updates and revisions for drivers that support their hardware.

Because electronic distribution is the most effective way to deliver the latest drivers and virtual devices to Windows customers, Microsoft distributes the WDL through the following online services:

- CompuServe
- GEnie
- Microsoft Product Support Services (PSS)
- PSS bulletin boards
- User Group bulletin boards

Distribution by disk through Microsoft fulfillment is also available for any customer without a modem.

You can add your device driver or virtual device to the WDL by passing the appropriate driver-testing requirements, and sending your driver or virtual device to Microsoft for final certification. The driver testing requirements are specified in the Printer Compatibility Test and Display Compatibility Test Suites provided with the respective test suite. For more information about Compatibility Tests, see Section 2.7, "Testing a Driver," in Chapter 2, "Producing Device Drivers and Virtual Devices."

There are no deadlines for participation in the WDL; new drivers can be added at any time. Microsoft releases drivers for electronic distribution immediately after they have passed compatibility tests at Microsoft. WDL fulfillment disks are manufactured once per quarter.

Getting Started

Chapter 1

This chapter explains what you need to begin developing Microsoft Windows device drivers and virtual devices. It describes the hardware and software you need, and explains how to install the Microsoft Windows Device Driver Kit (DDK).

1.1 What You Need to Get Started

You need a *development computer* on which you can edit, assemble, compile, and link sources for your driver or virtual device. You also need one or more *test computers* on which to test and debug your driver or virtual device. In some cases, you can both develop and test on your development computer. However, it is much more convenient, and in most cases necessary, to have separate development and test computers.

In all cases, your development computer should have the following minimum hardware configuration:

- A 386-based computer (with a clock speed of 16 MHz or above)
- At least 2 megabytes or more of memory (4 megabytes is recommended for 386 enhanced mode)
- A fast hard drive for your compiler or linker
- A fully configured EGA or VGA display card (or other device if you are developing a display driver and virtual-display device)
- A monochrome adapter and monitor, or a terminal connected to your AUX port
- A Microsoft Mouse or any other mouse compatible with Windows
- A Windows-compatible printer

Your development computer should also have the following software:

- Microsoft MS-DOS® version 5.0 or later
- Microsoft Windows 3.1
- Microsoft Windows 3.1 Software Development Kit (SDK)
- Microsoft Macro Assembler (MASM) version 5.10A or higher
- Microsoft Optimizing C Compiler version 6.0A or higher

Note Microsoft Macro Assembler version 5.10A is provided on the Device Driver Samples and Tools disk of the DDK. This version of MASM, needed to create drivers that match the retail Windows 3.1 drivers byte for byte, is no longer publicly available.

In all cases, you should install this software before installing the DDK. Carry out all changes to the CONFIG.SYS and AUTOEXEC.BAT files recommended by the respective installation programs.

Your test computers should have hardware and software configurations that are compatible with your device. You should test your driver or virtual device in all configurations you intend to support. Microsoft MS-DOS version 3.3 or later should be installed on all test computers.

1.2 Preparing for Installation

To prepare for a successful installation of the DDK, you need to create directories on your hard drive, set environment variables, and print a listing of the contents of the DDK disks.

1.2.1 Development Directories

Choose the directories that will receive the installed DDK tools, libraries, and header files. It is recommended that you use the WINDEV, WINDEV\LIB, and WINDEV\INCLUDE directories that you created when you installed the SDK. If you choose to use different directories, make sure you create those directories before starting the installation.

You also need to determine where to copy the directories containing the sample source files. It is recommended that you copy these sample directories to the WINDEV directory. For example, if you copy the keyboard driver samples to your hard drive, you'll create a WINDEV\KEYBOARD directory, and fill it with source files.

For virtual device samples, either copy the sample source directories to the same directory, or modify the header file sections in the sample makefiles. For example, if your INCLUDE directory is WINDEV\INCLUDE, copy the sample directories to WINDEV.

1.2.2 PATH, INCLUDE, and LIB Environment Variables

Some steps in the DDK installation require that you run the utilities you installed in a previous step. To ensure that the correct utility is run, add the WINDEV directory to your PATH environment variable before starting the installation. Since existing utilities on your hard drive may conflict with the DDK utilities, make sure the WINDEV directory is placed before any other directory, including

the MS-DOS directory. If you have directories containing tools from previous versions of the Windows DDK, it is a good idea to remove those directories from the PATH variable.

Although the INCLUDE and LIB environment variables are not used during installation, you should add the WINDEV\INCLUDE and WINDEV\LIB directories to them. Again, if you have directories containing header files or libraries from previous versions of the Windows DDK, it is a good idea to remove those directories from the INCLUDE and LIB environment variables.

Finally, you should make these changes permanent by modifying the MS-DOS **path** and **set** commands in your AUTOEXEC.BAT file.

1.2.3 Printing the LAYOUTS.TXT File

To help you locate the files you need to install, the LAYOUTS.TXT file on the Device Driver Samples and Tools Disk #1 specifies the directory structure of the DDK disks. Before starting the installation procedure, you should print this file, and refer to it while installing the DDK.

1.3 Installing the DDK

The DDK disks consist of device driver and virtual device sources, tools, utilities, and online information. You install most of these files by manually copying files from the DDK disks to directories on your hard drive.

► **To install the DDK:**

1. Locate the LIB directory on one of the Device Driver Samples and Tools disks, and copy all libraries to your WINDEV\LIB directory.
2. Locate the INC directory on one of the Device Driver Samples and Tools disks, and copy all header files to your WINDEV\INCLUDE directory.
3. Locate the TOOLS directory on one or more of the Device Driver Samples and Tools disks, and copy all executable files to your WINDEV directory.
4. Copy the following files to the WINDEV directory:

   EXPAND.EXE
   TREEX2.BAT
   TREEXP.BAT
   WALK.EXE

   These files are located in the Device Driver Samples and Tools Disk #1.

▶ To install virtual device files:

1. Locate the INCLUDE directory (on one of the Virtual Device Samples and Tools disks), and copy all header files to your WINDEV\INCLUDE directory.
2. Locate the TOOLS directory (on one or more of the Virtual Device Samples and Tools disks), and copy all executable files to the WINDEV directory.

▶ To install Windows device driver and virtual device sample sources:

1. Locate the directory or directories containing the sample you want on either the Device Driver Samples and Tools disks or the Virtual Device Samples and Tools disks.

   Make sure you check all disks for the appropriate source files—some source directories are too large to fit on a single disk.
2. Use the **xcopy** command to copy all sample files and directories to the WINDEV directory on your hard drive.

   For example, to install the sample sources for 1-plane display drivers, use the following command:

   ```
   xcopy a:\display\1plane c:\windev\display\1plane /s /e /v
   ```

   Use the **/s** switch to copy subdirectories, the **/e** switch to copy empty directories, and the **/v** switch to help you verify that the files are copied correctly.
3. Decompress the source files using the batch file TREEEXP.BAT now located in your WINDEV directory.

   To decompress files, start at the topmost directory first. For example, to decompress the 1-plane display driver sources, use the following commands:

   ```
   c:
   cd \windev\display\1plane
   c:\windev\treeexp
   ```

   The TREEEXP.BAT and related files decompress each file in the directory, and in all subdirectories.

▶ To install the minidriver samples and tools:

1. Copy EXPAND.EXE from the Minidriver Samples and Tools disk to the WINDEV directory on your hard drive.

2. Locate the INC directory on the Minidriver Samples and Tools disk, and expand and copy all header files to the WINDEV\INCLUDE directory on your hard drive. Use the following command:

```
expand a:\inc\*.* c:\inc
```

3. Locate the LIB directory on the Minidriver Samples and Tools disk, and expand and copy all library files to the WINDEV\LIB directory on your hard drive. Use the following command:

```
expand a:\lib\*.* c:\lib
```

4. Create a UNITOOL directory on your hard drive.
5. Locate the UNITOOL directory on the Minidriver Samples and Tools disk, and copy UNITOOL.EXE and UNITOOL.HLP to the WINDEV directory on your hard drive. These files are not compressed.
6. Locate the UNITOOL directory on the Minidriver Samples and Tools disk, and copy the UNITOOL.INF file to your WINDOWS directory. This file is not compressed.

Once UNITOOL.INF is installed, UNITOOL.EXE can automatically copy and expand the sample files it requires from the Minidriver Samples and Tools disk, so no additional installation is required for UniTool.

► To install the Multimedia device driver samples:

1. Insert the disk in a disk drive and, from the MS-DOS command line, change to that drive.
2. Type the command:

```
install c:\windev
```

3. Carry this procedure for each of the Multimedia Samples and Tools disks.

► To install Compatibility Tests and online documentation:

1. Install the online information files. For each Online Documentation disk, copy all files on the disk to the WINDEV directory on your hard drive.
2. Install the Display Compatibility Test Suite (DCT). Insert the Windows DCT Setup disk in a disk drive and, from the DOS command line, change to that drive. Type the command:

```
dctsetup
```

DCT automatically starts Windows when it starts. Follow the instructions on your screen.

3. Install the Printer Compatibility Test Suite (PCT). Insert the Windows PCT Setup disk in a disk drive and, from the MS-DOS command line, change to that drive. Type the command:

   ```
   pctsetup
   ```

   Follow the instructions on your screen.

4. Install the Network Compatibility Test Suite (NCT). Insert the NCT disk #1 in a disk drive and, from the MS-DOS command line, change to that drive. Type the command:

   ```
   install
   ```

   Follow the instructions on your screen.

1.4 Verifying the Installation

Make sure all tools and sources you need have been properly installed. The best way to do this is to build one of the sample device drivers.

► **To verify the installation:**

1. Close all applications and remove all memory-resident programs (as needed) to free 620K of conventional memory. If necessary, load device drivers in high memory.
2. Choose a sample driver to build.

   In general, choose a driver that supports the same type of device as yours.
3. Check the makefile of the sample driver for any build instructions located at the beginning of the file.
4. Use the Make program to build the sample driver. Make compiles, assembles, and links driver sources as directed by the makefile.

The Make program displays an error message and stops if the necessary tools or sources are not properly installed. If a error occurs, review the error message and make sure the specified tool or source file is where it should be.

Producing Device Drivers and Virtual Devices

Chapter 2

This chapter describes how to create and distribute a device driver or virtual device. The work you do depends on whether your device is compatible with those supported by Microsoft Windows, whether the device is accessible to MS-DOS programs, and whether you have an existing Windows device driver or virtual device.

2.1 When Do You Need a Device Driver or Virtual Device?

You need a *device driver* if you intend to give Windows or Windows applications access to your device, and the device driver is not 100 percent compatible with similar devices supported by the retail Windows drivers. You will also need a device driver if your device has extended features not supported by the Windows drivers.

You need a *virtual device* if you intend to give MS-DOS programs access to your device while Windows runs in 386 enhanced mode. If you provide an MS-DOS installable device driver or memory-resident program that gives MS-DOS programs direct or indirect access to your device, you need to create a virtual device. If your device generates interrupts that are not serviced by the ROM BIOS or if your device relies on a nonstandard ROM BIOS, you need a virtual device.

► **To create a device driver or virtual device:**

1. Write new sources, or modify the sample sources.
2. Choose the filename and module name for your driver, and an ID number for your virtual device.
3. Compile, assemble, and link.
4. Add a version stamp to your driver files.
5. Debug your device driver or virtual device.
6. Test your device driver or virtual device.
7. Create your OEMSETUP.INF file and distribution disks.
8. Test the distribution disks by using Windows Setup or Control Panel to install your driver or virtual device.

In all cases, it is recommended that you start with an existing driver or virtual device. If you do not already have a driver or virtual device, review the sample drivers provided in the DDK and start with a sample that supports a device that matches your device's capabilities.

To prevent maintenance and upgrade problems, vendors must rely on Windows Setup or Control Panel to install their device drivers and virtual devices. This means you must create an OEMSETUP.INF file for your driver disks that contains the instructions Setup or Control Panel need to install your files. Also, you must clearly instruct your customers to use Windows Setup or Control Panel to install new drivers.

Important Never write installation programs to install Windows drivers.

2.2 Adapting an Existing Driver

Windows 3.1 is fully compatible with most Windows 3.0 device drivers and virtual devices. In most cases, you can continue to distribute your existing driver for use with both Windows 3.0 and 3.1. For some drivers, such as display drivers that utilize font caching, you may need to modify the driver to work properly with Windows 3.1.

To derive the greatest benefit from Windows 3.1, modify your existing driver to incorporate Windows 3.1 features. Modify your driver such that it continues to work properly with Windows 3.0 and 3.1 so that you need maintain only one version of your driver. In all cases, you must thoroughly test your driver with Windows 3.0 and 3.1 and correct any problems.

Although Windows 3.1 does not support real mode, if you create a driver that runs with both Windows 3.0 and 3.1, your driver will need to be a bimodal driver. For more information about bimodal drivers, see the Microsoft Windows 3.0 Device Driver Kit.

If you update your driver, feel free to change description strings to indicate the update, but leave the filename the same as the existing driver. This allows for faster support by Microsoft and your own technical support personnel. In particular, if you are licensing driver files, do not change the driver filenames unnecessarily.

2.3 Modifying Sample Drivers

The sample drivers provide complete sources for creating many of the retail Windows 3.1 drivers. Using a sample driver as the starting point for your own driver can save you considerable time in incorporating the Windows 3.1 features.

If you decide to modify a sample driver, make sure you install all source files, tools, and related files from the DDK disks. If you change the organization of the sample files on your hard disk, be sure to make the appropriate changes to the make or batch file for building the driver.

2.4 Choosing Names and IDs

To prevent maintenance and upgrade problems, carefully choose the file and module names for your driver, and the filename and ID number for your virtual device.

If you create your own device driver, even if it is a slight modification of a sample driver, you must provide a unique filename for the driver. The module name should be as given in the following list.

Driver	Module name
Communications	COMM.
Display	DISPLAY.
Grabber (standard mode)	No module name required.
Grabber (386 enhanced mode)	GRABBER.
Installable Drivers	Must be unique.
Keyboard	KEYBOARD.
Logo code	No module name required.
Logo data	No module name required.
Mouse	MOUSE.
Network	Must be unique.
Printer	Must be unique.
Sound	SOUND.

If your virtual devices replace standard virtual devices included with Windows 3.1, use the same virtual device ID numbers as the devices being replaced. If your virtual device is a supplement, rather than a replacement, you may need to obtain a virtual device ID number from Microsoft DDK Product Support Services (PSS). Virtual device ID numbers are assigned by the DDK PSS. There are a number of ways to request virtual device IDs:

- Microsoft OnLine Service.
- Microsoft SDK Forum on CompuServe.
- Electronic mail to vxdid@microsoft.com through MCI MAIL, CompuServe, or any service that has InterNet access.

If sending electronic mail, you may need a prefix for a given service, for example, CompuServe mail is: >INTERNET:vxdid@microsoft.com.

In all cases, if you create your own virtual device, even if it is a slight modification of a sample virtual device, you must give the virtual device a unique filename.

For more information about filenames for device drivers and virtual devices, see Appendix B, "Windows 3.1 Driver Filename Policy."

2.5 Adding a Version Stamp

You add a **VERSIONINFO** resource to your device driver's executable file or files to help Windows Setup or Control Panel distinguish your files from other files with the same name (particularly previous versions of your driver). Windows Setup uses the resource to determine whether it should install your files if files having the same name already exist on the user's hard drive. The Microsoft Windows 3.1 Software Development Kit provides instructions on how to add a version resource to your Windows executable files.

2.6 Debugging a Driver

You can debug the driver by using one of the following debugging tools.

Tool	When Used
CodeView® for Windows (CVW)	Used mainly for debugging drivers written in the C language. CVW requires appropriate hardware, such as a secondary monitor, as well as appropriate compiler and link options to generate symbolic information.
WDEB386	Used for debugging drivers written in the assembly language. WDEB386 is required for debugging virtual devices in 386 enhanced mode. Despite its name, WDEB386 can be used with the 80286, 80386, or 80486 processor.

WDEB386 uses the first serial port (COM1) for all debugging input and output. This means you must connect a terminal (or a computer running terminal software) to COM1. WDEB386 sends and receives data according to the following communications specifications:

- 9600 baud rate
- No parity
- 8 bits per character
- 1 stop bit

Specify symbol files for the driver you intend to debug, and any drivers or system modules that the driver may call. The symbol files contain the information WDEB386 needs to display the names of the functions, variables, and constants in the driver.

2.7 Testing a Driver

You use the display, printer, and network Compatibility Test Suites to verify the performance and operation of your device drivers and to satisfy the requirements of Driver Compatibility Testing Program. Meeting these requirements qualifies your drivers for certification and inclusion in the Microsoft Windows Driver Library (WDL).

2.7.1 Display Driver Compatibility Tests

The Display Driver Compatibility Test (DCT) verifies the robustness and correct operation of a display driver and its supporting virtual display device, standard-mode grabber, and 386 enhanced-mode grabber. You can run the tests as stand-alone tools during the development and debugging stages of your driver, or you can start the tests using the CT shell. The CT shell is a Windows application that starts the tests, keeps track of test results, and provides an onscreen report.

For a complete description of the tests and an explanation of how to use them, see the DCT.DOC file that you installed with the DCT.

2.7.2 Printer Driver Compatibility Tests

The Printer Driver Compatibility Test (PCT) verifies the robustness and correct operation of a printer driver. The PCT contains standard tests and tools that all printer driver developers use to help improve the quality of Windows printer drivers. For a complete description of the tests and an explanation of how to use them, see the PCT.DOC file that you installed with the PCT.

2.7.3 Network Driver Compatibility Tests

The Network Compatibility Test (NCT) verifies the compatibility between networking environments and Microsoft Windows version 3.1. The NCT shell is a Windows application which starts the tests, keeps track of test results, and provides an onscreen report. You can also use several tests as standalone test tools during the development and debugging stages of your driver. For a complete description of the tests and an explanation of how to use them, see the NCT.DOC file that you installed with the NCT.

2.8 Creating Distribution Disks

Once you have thoroughly tested and debugged your device driver and virtual devices, create the distribution disk or disks from which users can install the driver.

Before you create a distribution disk, you must create an OEMSETUP.INF file. This file specifies the files and initialization file settings that need to be installed to support your driver. The OEMSETUP.INF file consists of one or more sections, depending on the type of driver.

Driver	Required	Optional
Display	[disks] [display] [oemfonts] [sysfonts] [fixedfonts] [fonts] [*386Grabber*]	[*AdditionalInfo*]
Pointing device	[disks] [pointing.device]	[*AdditionalInfo*]
Keyboard	[disks] [keyboard.types]	[data] [keyboard.tables]
Network	[disks] [network]	[*WinIniSect*] [*SysIniSect*] [*Netname*.versions] [*work-section*]
Printer	[disks] [io.device]	[io.dependent]

For more information about the OEMSETUP.INF file, see *Microsoft Windows Device Driver Adaptation Guide*.

▶ **To create the distribution disk:**

1. Copy the OEMSETUP.INF file to the distribution disk.

   Do not compress this file.

2. Copy all required files to the distribution disks.

   This includes any files from the Windows 3.1 retail Setup disks that your driver may require. As you copy these files, compress them using the COMPRESS program provided in the Microsoft Windows 3.1 Software Development Kit.

You must *not* require the user to use the Windows 3.1 retail disks to install files needed by your driver. Always provide all files needed for installation on your distribution disk. If you supply the files from the retail disks, always use the same filenames for these files as used by Windows. Proper maintenance and upgrades depend on preserving these filenames.

Make sure you provide explicit, step-by-step instructions for your customers to install *all* files (including MS-DOS device drivers and utilities) necessary to use your device with Windows.

If you create a distribution disk for a display driver, you need to copy files having the following extensions: DRV, FON, 3GR, 2GR, 386, LGO, and RLE. Never reference *vddvga in your OEMSETUP.INF file. If your driver requires the virtual display device for VGA, you must build and include the VDDVGA.386 file on your distribution disk.

For PostScript printers, Windows now installs the Windows Printer Description (WPD) files just like any other printer driver in Windows. For an example of what you need to add to the OEMSETUP.INF file for a WPD file, see the CONTROL.INF file in the Windows SYSTEM directory. For more information about the OEMSETUP.INF file, see the *Microsoft Windows Printers and Fonts Kit*.

2.9 Creating a Combined Windows 3.0 and 3.1 Distribution Disk

If you have separate drivers for Windows 3.0 and 3.1, you can distribute both drivers on the same distribution disk if you clearly separate the drivers and related files, and provide a separate OEMSETUP.INF file for each.

► **To create a combined distribution disk:**

1. Create a WIN30DRV directory and copy all files that are specific to Windows 3.0 to this directory.
2. Create a WIN31DRV directory and copy all files that are specific to Windows 3.1 to the directory.
3. Clearly instruct users to specify the appropriate directory when Setup prompts them for the OEM Driver Disk. For example, when installing Windows 3.1, and the disk is in drive A, direct the user to type:

```
a:\win31drv
```

What's New for Windows Version 3.1?

Chapter 3

This chapter describes the device driver and virtual device features that are new for Microsoft Windows 3.1.

3.1 Multimedia Device Drivers

You can enrich Windows with audio, Media Control Interface (MCI), and joystick capabilities by creating Windows drivers for these multimedia devices. The audio, MCI, and joystick interfaces available through Windows extends the environment by offering a standard set of devices and capabilities. By developing multimedia device drivers, you can customize and extend the system environment to include your specific devices. The samples for multimedia devices include driver sources, testing and installation applications, header files, and documentation describing the key features of the drivers.

3.2 UniTool and Sample Minidrivers

The Microsoft Windows Universal Printer Driver eases the effort to create and maintain printer drivers for raster printers for the Windows environment. The underlying principle of the Universal Printer Driver is to separate the printer-specific information from the Windows-specific information. Each printer will have a separate file that contains information on the capabilities of the printer. All executable driver code will reside in a separate common code library known as the Windows Universal Printer Driver.

Supporting a printer using the Universal Printer Driver Architecture requires the creation of a "minidriver." The minidriver is a binary file that is created using the Windows-based printer-driver development tool UniTool. The minidriver file contains all of the printer-specific information for the printer. This information is all printer hardware oriented. It includes information such as supported paper sizes, font cartridges available, imageable page area, cursor-movement commands, and so on. It contains no Windows-specific information. The minidriver can optionally contain code to handle special features or problems with a specific printer.

The Universal Printer Driver for Windows 3.1 (UNIDRV.DLL) contains all of the traditional code for translating output from the Windows Graphics Device Interface (GDI) imaging model to the printer's specific text and graphics functions. The Universal Printer Driver relies on the information within the printer's minidriver to determine the basic technology of the printer and what its specific capabilities are.

For more information about creating minidrivers, see the *Microsoft Windows Minidriver Development Guide*.

3.3 Communications Drivers

This section describes the new features for the communications driver.

3.3.1 Base Address and IRQ Selection

The Windows 3.1 communications driver (COMM.DRV) accesses serial ports COM1, COM2, COM3, and COM4 using base address values and interrupt request lines (IRQs) specified in the BIOS data area of a computer. If the BIOS data area does not specify values for physical ports, Windows 3.1 will use the values specified by the [386Enh] section of SYSTEM.INI.

To override these defaults, the user can set the base address and IRQ values using the Control Panel. Control Panel displays an "advanced" settings dialog box for each port. The dialog box contains drop-down edit fields for base addresses and spin controls for IRQ settings. The selected values are recorded as **COM*x*Base** and **COM*x*IRQ** settings in the [386Enh] section of the SYSTEM.INI file. Note that these settings are used for both standard and 386 enhanced-mode operation.

3.3.2 16550 UART FIFO Buffer

If a computer uses the 16550 Universal Asynchronous Receiver Transmitter (UART) for its communication ports, the communications driver will enable the onboard 16-byte First In, First Out (FIFO) buffer allowing Windows to perform reliable serial communications at speeds of 9600 baud and higher. (Many systems that do not enable this buffer experience loss of characters at 9600 baud, and most cannot communicate at speeds higher than this.)

Before enabling the FIFO buffer, the communications driver checks the SYSTEM.INI file to determine whether the user wants the buffers enabled. The **COM*x*FIFO** settings in the [386Enh] section of the SYSTEM.INI file specify whether the buffer for a given port should be enabled or disabled. If a setting should be enabled, the driver enables the FIFO buffer; otherwise, it disables the buffer. If no setting is specified, the driver will enable the buffer by default

3.3.3 The CommWriteString and EnableNotification Functions

The communications driver exports the **CommWriteString** and **EnableNotification** functions. The **CommWriteString** function writes a string of one or more bytes to the given communications device. The function improves performance by eliminating the series of individual calls to the **sndcom** function found in earlier versions of Windows.

The **EnableNotification** function enables or disables port status notifications. If notifications are enabled, the communications driver posts WM_COMMNOTIFY messages to a given window at the occurrence of certain events. This eliminates the need for communications applications to set timers and use the **GetCommError**, **GetCommEventMask**, and **SetCommEventMask** functions to watch for port status changes. This can result in system-wide performance improvements since the communications application no longer needs to service connections that may be inactive or take exclusive use of a system timer.

3.3.4 Extended Escapes

The communications driver supports the RESETDEVICE, GETBASEIRQ, GETMAXLPT, and GETMAXCOM extended escapes in its **cextfcn** function. These extended escapes reset the printer (assert the reset line), and retrieve the maximum number of identifiers for parallel and serial ports.

3.3.5 Baud Rate Indexes

The communications driver supports very high baud rates, such as 128,000 and 57600, by interpreting the **BaudRate** member of the **DCB** structure as a baud-rate index whenever the high byte of the member is 0xFF. The driver that ships with the DDK has a maximum baud rate of 57600. In such cases, **BaudRate** can be one of the following values.

Value	Baud rate
CBR_110 (0xFF10)	110
CBR_300 (0xFF11)	300
CBR_600 (0xFF12)	600
CBR_1200 (0xFF13)	1200
CBR_2400 (0xFF14)	2400
CBR_4800 (0xFF15)	4800
CBR_9600 (0xFF16)	9600

Value	Baud rate
CBR_14400 (0xFF17)	14400
CBR_19200 (0xFF18)	19200
CBR_38400 (0xFF1B)	38400
CBR_56000 (0xFF1F)	56000
CBR_128000 (0xFF23)	128,000
CBR_256000 (0xFF27)	256,000

Note The CBR_ values are for standardization; drivers are not required to support all indexed baud rates.

If the high byte of the **BaudRate** member is not 0xFF, the **BaudRate** member specifies the actual baud rate for the communications device. In other words, values in the range 2 through 65,279 (0xFEFF) are interpreted as baud-rate values, not as indexes. This ensures compatibility with existing communications drivers.

3.4 Display Drivers

Microsoft has made every effort to preserve compatibility of Windows 3.0 display drivers with Windows 3.1. However, there are some basic guidelines that must be followed to ensure full compatibility.

3.4.1 Font Caching for TrueType Fonts

Display drivers that do their own font caching may encounter conflicts with the new TrueType font technology provided with Windows 3.1. Display drivers that use glyph caching, rather than caching the entire character set, should work without problems. The 8514/a driver source code provides a sample of a driver that uses glyph caching.

3.4.2 Virtual Display Devices and Screen Grabbers

Many third-party display drivers utilize virtual-device displays (VDD) and grabbers included in Windows 3.0. Combinations of Windows 3.0 VDDs and grabbers with their counterpart components included in Windows 3.1 may not be compatible. For example, a grabber designed to work with the standard VGA VDD included in Windows 3.0 (*VDDVGA) may not be compatible with the standard VGA VDD included in Windows 3.1.

In all cases for display drivers, whether it's Windows 3.0 or 3.1, you must create and distribute a Setup disk that contains VDDs and grabber files. Don't depend on the Windows retail product Setup disks for the distribution of these files.

Properly built third-party display driver OEM Setup disks will already include appropriate VDD and grabber files, and no changes should be required. Unfortunately, some vendors do not ship VDD and grabber files that are already available on the Windows 3.0 Setup disks.

If your driver installation disk provides a proprietary VDD, but relies on standard Windows 3.0 disks for a grabber, you should rebuild your installation disks to ensure compatibility with Windows 3.1.

Similarly, if your driver installation disk provides a proprietary grabber, but relies on standard Windows 3.0 disks for a VDD, you should rebuild your installation disks to ensure compatibility with Windows 3.1.

The driver installation disks must include all necessary files. You can determine the necessary files from the table below. In general, if your driver uses the specified Windows 3.0 file, your Setup disks should include the corresponding Windows 3.1 file.

Windows 3.0 file	Windows 3.1 file	Description
*VDDVGA	VDDVGA30.386	Windows 3.0 VGA virtual-display device. This file is equivalent to the original VGA VDD embedded in the Windows 3.0 WIN386.EXE file. It should be used only with compatible 386 enhanced-mode grabbers, such as VGA30.3GR.
VGA.GR3	VGA30.3GR	Windows 3.0 VGA 386 enhanced-mode grabber. This file is equivalent to the original Windows 3.0 VGA 386 enhanced-mode grabber. It should be used only with compatible VDDs, such as VDDVGA30.386.
VDD8514.386	VDD8514A.386	Windows 3.0 8514/a virtual-display device. This file is built from the Windows 3.1 EGA VDD source files, and contains minor updates since the release of the original Windows 3.0 8514/a VDD. It should be used only with compatible 386 enhanced-mode grabbers, such as V7V6A.3GR.

Windows 3.0 file	Windows 3.1 file	Description
8514.GR3	V7V6A.3GR	Windows 3.0 V7V6A 386 enhanced-mode grabber. This file is equivalent to the original Windows 3.0 8514/a 386 enhanced-mode grabber. It should be used only with compatible VDDs, such as VDD8514A.386.

The Windows 3.1 DDK disks include Windows 3.0 versions of VDDs and grabbers previously described in the DISPLAY\WIN30 directory.

VGACOLOR.2GR, the Windows 3.1 standard-mode VGA color grabber, is compatible with drivers that use the Windows 3.0 version, VGACOLOR.GR2.

V7VGA.3GR, the Windows 3.1 version of the Video Seven 386 enhanced-mode grabber, is compatible with drivers that use the Windows 3.0 version, V7VGA.GR3.

3.4.3 Device Bitmaps

A device bitmap is any bitmap with bits that are stored in device memory (such as RAM on a display adapter) instead of main memory. Device bitmaps can significantly increase the performance of a graphics driver as well as free main memory for other uses. To realize these benefits, the corresponding graphics device must have ample video memory in addition to the video memory used to generate the current display. The device should also have efficient routines for copying bits to video memory.

Graphics drivers that set the RC_DEVBITS bit in the **dpRaster** member of the **GDIINFO** structure support device bitmaps. GDI checks this bit to determine how to carry out requests to create and select bitmaps. If a driver sets the RC_DEVBITS bit, it must export the following functions:

BitmapBits
RealizeObject
SelectBitmap

The **BitmapBits** function copies bitmap data to and from device bitmaps. GDI calls this function when initializing the bits after creating the bitmap. It also calls the function when an application calls the functions such as **GetBitmapBits** and **SetBitmapBits**.

The **RealizeObject** function creates or deletes a device bitmap. GDI calls this function when creating the bitmap by specifying the OBJ_BITMAP style. **RealizeObject** is responsible for allocating memory for the device as well as filling a physical **PBITMAP** structure that GDI uses to identify the device bitmap. If the bitmap is to be deleted, **RealizeObject** must free the device memory.

The **SelectBitmap** function associates a device bitmap with the given **PDEVICE** structure. GDI passes the physical **PBITMAP** structures of both the currently selected bitmap and the new bitmap so that **SelectBitmap** can carry out any special processing to enable or disable access to the device bitmaps.

Device bitmaps can not be monochrome bitmaps. GDI intercepts all requests to create monochrome bitmaps, and creates main memory bitmaps instead. This means a graphics driver that supports device bitmaps must also support main memory bitmaps.

3.4.4 Transparent Block Transfers

In Windows 3.1, display drivers can indicate that they support transparent block transfers by setting the C1_TRANSPARENT bit in the **dpCaps1** member of the **GDIINFO** structure. In a transparent block transfer, a driver excludes source and brush pixels from a **BitBlt** or **StretchBlt** operation if those pixels have the same color as the current background color for the destination device.

If a display driver supports transparent block transfers, the **BitBlt** function must check the **bkMode** member of the *lpDrawMode* parameter as well as the *Rop3* parameter to determine how to carry out the transfer. If the **bkMode** member specifies the background mode TRANSPARENT1, **BitBlt** must not transfer source and brush bits that have the same color as the destination's background color. In other words, the corresponding destination bits must be left unchanged. Other background modes do not affect the transfer.

3.4.5 Large Icons and Cursors

In Windows 3.1, display drivers can use icons larger than 64-by-64 bits and cursors larger than 32-by-32 bits. Large icons and cursors can improve screen readability for high-resolution graphics adapters.

Display drivers specify icon and cursor size in the **IconXRatio**, **IconYRatio**, **CurXRatio**, and **CurYRatio** members of the CONFIG.BIN resource (that is, the resource having identifier 1 and type OEMBIN). In Windows 3.1, these members specify either a width and height in pixels, or a compression ratio.

In Windows 3.0, these members only specify compression ratios. For more information about the members, see the *Microsoft Windows Device Driver Adaptation Guide.*

In all cases, each icon or cursor must have the same width and height (measured in pixels).

Display drivers which specify actual widths and heights in the OEMBIN resource cannot be used with Windows 3.0. Drivers which specify compression factors work with both Windows 3.0 and 3.1. However, drivers cannot simultaneously specify compression factors with icons or cursors larger than 32-by-32 bits.

3.4.6 Default System Colors

The CONFIG.BIN resource specifies the default system colors (as well as several other OEM-specific values). For Windows 3.1, the recommended default system color values for 16- and 256-color displays are new. In particular, several system colors that were patterned (dithered) in Windows 3.0 have been changed to solids to improve performance and visibility for interlaced displays.

In Windows 3.1, the CONFIG.BIN resource has been expanded to include the **clrInactiveCaptionText** member. This member specifies the color of the text in the title bar of an inactive window.

For more information and a complete description of the default system colors and the **clrInactiveCaptionText** member, see the *Microsoft Windows Device Driver Adaptation Guide.*

3.4.7 Grabber Extensions

Windows 3.1 grabber file extensions have been changed to .2GR for standard-mode grabbers, and .3GR for 386 enhanced-mode grabbers. This change should not affect the operation or installation of existing display drivers with Windows 3.1.

3.4.8 Color Palette Management

Graphics drivers that support color palettes must make sure that the index for the palette entry that corresponds to black must be the one's complement of the index for the palette entry for white. Black and white must be "static" palette entries. This means the driver sets the indexes for these colors during initialization, and does not change the indexes.

3.4.9 Extensions to ANSI Character Set

Windows 3.1 extends the ANSI character set for TrueType fonts. The Windows ANSI character set now includes new characters for TrueType fonts only.

ANSI position	Character
130 (0x82)	Baseline (single quote).
131 (0x83)	Florin (latin small letter script f).
132 (0x84)	Based Line Double Quotes (low double comma quotation mark).
133 (0x85)	Ellipsis (horizontal ellipsis).
134 (0x86)	Dagger (Olelisk).
135 (0x87)	Double Dagger.
137 (0x89)	Permil (per mille sign).
138 (0x8A)	S Hacek (uppercase; latin capital letter s hacek).
139 (0x8B)	Diminutive Less Than Sign (left pointing single guillemet).
140 (0x8C)	Ligature (uppercase; latin capital letter o e).
147 (0x93)	Open Double Quotes (double turned comma quotation mark).
148 (0x94)	Close Double Quotes (double comma quotation mark).
149 (0x95)	Bullet.
150 (0x96)	Short En Dash (en dash).
151 (0x97)	Long Em Dash (em dash).
152 (0x98)	Tilde.
153 (0x99)	Trademark Symbol.
154 (0x9A)	s Hacek (lowercase; latin small letter s hacek).
155 (0x9B)	Diminutive Greater Than sign (right pointing single guillemet).
156 (0x9C)	Ligature (lowercase; latin small letter o e).
159 (0x9F)	Y Dieresis (uppercase; latin capital letter y diaresis).

3.4.10 Multiple-Resolution Drivers

A multiple-resolution display driver provides the code and resources needed to support all resolutions of a given display device. In previous versions of Windows, separate display drivers were required for each resolution. In Windows 3.1, a single display driver can handle all resolutions.

A multiple-resolution driver must provide the following:

- Icons and cursors for each supported resolution
- **GetDriverResourceID** function
- Installation information

3.4.10.1 Resources and Resource Mapping

In many cases, the only difference between the device resolutions for the driver is the size of the resources and the horizontal and vertical dimensions. A driver can support multiple resolutions as long as it can determine which resolution Windows expects it to use.

To support multiple resolutions in a single display driver, Windows checks for the **GetDriverResourceID** function in the driver, and calls it before loading the driver's resources and dimensions. This gives the driver a chance to check the SYSTEM.INI file for information specifying the desired resolution. The **GetDriverResourceID** function can then map the request resource identifier to the identifier of the corresponding resource for the desired resolution. In this way, the driver makes sure Windows loads the appropriate resources for the selected resolution.

Display drivers that support multiple resolutions must export the **GetDriverResourceID** function and provide a set of appropriate resources for each resolution.

3.4.10.2 Installation Information

Users select the desired resolution for the display using the Setup program. To display each screen resolution, Setup checks the [display] section in the SETUP.INF or OEMSETUP.INF file. There must be one line for each resolution. For example, if a video adapater named "ZGA" has three resolutions (640x480, 800x600, and 1024x768), the [display] section should look something like this:

```
[display]
zga1=1:zga.drv,"ZGA (640x480)","100,96,96",3:zgacolor.2gr,..., zgalo
zga2=1:zga.drv,"ZGA (800x600)","100,96,96",3:zgacolor.2gr,..., zgamed
zga3=1:zga.drv,"ZGA (1024x768)","100,120,120",3:zgacolor.2gr,..., zgahi
```

Setup installs the same driver (ZGA.DRV), grabbers, virtual display device, and other related components. However, the last field in the line specifies an additional section in the SETUP.INF or OEMSETUP.INF file that contains resolution-specific information.

The resolution-specific information allows Windows Setup to copy files and write profile data for later use by the display driver. Setup can write the information to the SYSTEM.INI file in the section and format understood by the driver. When a display driver is initialized, it can use the Windows functions **GetPrivateProfileInt** or **GetPrivateProfileString** to read the user-specified screen resolution from the appropriate section of SYSTEM.INI. This will let the driver decide what resource ID to return to Windows by the **GetDriverResourceID** function.

The following examples show how resolution-specific information sections might look in the SETUP.INF or OEMSETUP.INF file:

```
[zgalo]
4:zgalo.dll,0:system,,,,

[zgamed]
,,system.ini,zga.drv,,"ZGAres=800x600"

[zgahi]
4:zgahi.dll,0:system,system.ini,zga.drv,"ZGAres=800x600","ZGAMode=Hi"
```

For more information about the resolution-specific information sections, see the *Microsoft Windows Device Driver Adaptation Guide*.

3.4.11 Mouse Trails

In Windows 3.1, display drivers can improve mouse cursor visibility by supporting mouse trails. A mouse trail is a sequence of two or more cursor images that mark current and previous mouse cursor positions. A display driver creates a mouse trail by drawing a cursor at each new mouse position and leaving additional cursors at previous positions. The driver delays erasing the previous cursors until a specified number of cursors are visible.

A display driver provides support for mouse trails by processing the MOUSETRAILS escape in its **Control** function. The MOUSETRAILS escape enables or disables mouse trails. It also sets the maximum number of cursors to be displayed in the trail. The display driver must also process the QUERYESCSUPPORT escape, returning the status of the mouse trails if the requested escape is MOUSETRAILS.

The display driver draws the mouse trail when it processes the **MoveCursor** function. Any trailing cursors must have the same shape as the current cursor. The display driver should provide mouse trails for system cursors, such as the pointer, I-beam, and hourglass, and for application-specific cursor as well.

The display driver is responsible for recording the mouse trail status by maintaining the **MouseTrails** setting in the [Windows] section of the WIN.INI file. The driver sets this setting to a positive value to indicate that the mouse trails are enabled. The value also specifies the number of cursors to display. The driver sets this setting to a negative value when the mouse trails are disabled. The absolute value specifies the number of cursors to display prior to being disabled.

Users can turn mouse trails on or off by using the Windows 3.1 Control Panel. The mouse settings dialog box displays a Mouse Trails option.

Applications should not enable or disable mouse trails. However, third-party mouse configuration programs should enable or disable mouse trails.

3.4.12 Optimizing Performance

Whenever possible, display drivers should use the features of the CPU to optimize performance. This is particularly important for drivers used in multimedia versions of Windows.

A display driver can determine what CPU is present and in what mode Windows is operating by examining the **__WinFlags** variable or by calling the **GetWinFlags** function. If the CPU is a 386 or 486, the display driver should take advantage of the CPU's 32-bit registers to manipulate data and to index huge arrays.

3.4.13 Pen-Capable Display Drivers

The DDK VGA driver source provides support for a Pen-Capable VGA display. To build a Pen-Capable VGA driver, you must set the PENWIN environment variable to 1.

Display drivers that support Windows for Pen Computing provide a set of functions and resources that permit the display drivers to carry out inking. Windows for Pen Computing is a version of Windows in which character-recognition software (the RC Manager) allows a pen device to be used in place of a keyboard. Inking is drawing done by a display driver in response to input from the RC Manager.

3.4.13.1 Inking Functions

Display drivers that support inking must be prepared to process inking requests whenever pen input generates an interrupt. The drivers must export the following functions:

GetLPDevice
InkReady

If the RC Manager requires inking, it calls the **GetLPDevice** function to retrieve a pointer to the display driver's physical device structure (**PDEVICE**). The RC Manager uses this pointer in subsequent calls to the display driver's **Output** function to complete the inking. The color and width of the ink is set by the RC Manager.

Before calling **Output**, the RC Manager calls the **InkReady** function to notify the display driver that it is ready to ink. The display driver must determine whether any other drawing operation is under way. If so, the display driver must wait until the current operation is complete before starting the inking. In any case, the display driver calls a callback function supplied with the call to **InkReady** to start the inking.

3.4.13.2 Inking Resources

Display drivers that support inking must provide the following cursor resources.

Value	Use
IDC_NEPEN (32630)	Pen points to northeast.
IDC_NWPEN (32631)	Pen points to northwest.
IDC_SEPEN (32632)	Pen points to southeast.
IDC_SWPEN (32633)	Pen points to southwest.
IDC_PEN (32633)	Default pen is same as IDC_SWPEN.

The pen cursors must be added to the driver in the same way as standard Windows cursors.

3.4.14 TrueType for Displays

Display drivers that handle raster fonts can also handle TrueType fonts without modification. GDI supports TrueType in display drivers by building **FONTINFO** structures that contain rasterized glyph bitmaps. GDI passes a pointer to this structure to the driver's **ExtTextOut** or **StrBlt** function along with the string to be displayed.

Display drivers can handle text in TrueType fonts almost identically to text in regular raster fonts. One important difference, however, is that TrueType glyphs are designed to be more readable by minimizing the gaps between glyphs—in fact, some glyphs overlap. To take full advantage of the design and maintain performance, display drivers may need some modification to allow for overlapping glyphs.

3.4.15 Sample Driver: Super VGA

Windows 3.1 includes a Super VGA driver providing screen resolution of 800x600 at 16 colors. This driver is based on the VGA driver source code.

This driver supports the 16-color 800x600 modes of most Super VGA display adapters, and is capable of automatically detecting many OEM numbers.

When the Super VGA driver loads, it checks the [SUPERVGA.DRV] section of the SYSTEM.INI file for a mode number specified by the svgamode setting. The setting takes a single decimal value specifying the video mode number. If the driver finds a value in SYSTEM.INI, it attempts to put the adapter into that mode.

If the mode switch fails, or if a value is not specified in SYSTEM.INI, the Super VGA driver will attempt to automatically detect the correct mode number. If it detects one, it writes this value out to SYSTEM.INI. If it does not, the driver fails to load, and the user is returned to the MS-DOS prompt.

Users with adapters not detected by the Super VGA driver must specify the correct (decimal) mode number in the [display] section of SYSTEM.INI.

Microsoft recommends that current users of 16-color 800x600 drivers switch to the driver included in Windows 3.1, since it contains new features and performance optimizations.

3.5 Printer Drivers

Windows 3.1 includes several new features that enhance the Windows printing environment.

3.5.1 Indexes for DeviceCapabilities Function

Printer drivers must process requests from applications for new **DeviceCapabilities** indexes. A driver's **DeviceCapabilities** function must check for and process the following values.

Value	Meaning
DC_ENUMRESOLUTIONS (13)	Retrieves a list of resolutions supported by the printer model.
DC_FILEDEPENDENCIES (14)	Retrieves a list of filenames which also need to be installed when the driver is installed.
DC_TRUETYPE (15)	Retrieves the driver's capabilities with regard to printing TrueType fonts.
DC_PAPERNAMES (16)	Enumerates the actual string describing the names of the papers.
DC_ORIENTATION (17)	Retrieves the relationship between Portrait and Landscape orientations in terms of the number of degrees that portrait orientation is to be rotated counterclockwise to achieve landscape orientation.
DC_COPIES (18)	Retrieves the maximum number of copies the device can print.

The implementation of one or all of the indexes is optional. If a driver does not implement a given index, the **DeviceCapabilities** function should return –1. For more information about the new indexes, see the **DeviceCapabilities** function in the *Microsoft Windows Device Driver Adaptation Guide*.

3.5.2 New Paper Sizes

There are 21 new predefined paper sizes. The include file, PRINT.H, contains a complete list of all supported paper sizes. Applications typically call the **DeviceCapabilities** function to retrieve names and sizes of the supported paper sizes, so printer drivers must be ready to process the following **DeviceCapabilities** index values:

Value	Meaning
DC_PAPER	Retrieves the identifiers of the listed paper sizes.
DC_PAPERSIZES	Retrieves the width and height of the listed paper sizes.
DC_PAPERNAMES	Retrieves the names of the listed paper sizes.

3.5.3 New Members in the DEVMODE Structure

The **dmYResolution** and **dmTTOption** members in the **DEVMODE** structure are new for Windows 3.1. The **DEVMODE** structure returned by the **ExtDeviceMode** function contains one or both of these members if the **dmField** member includes the following values:

```
DM_YRESOLUTION 0x0002000L
DM_TTOPTION    0x0004000L
```

The **dmYResolution** member specifies the vertical resolution of the printer in dots per inch. In this case, the **dmPrintQuality** member specifies the horizontal resolution in dots per inch. If DM_YRESOLUTION bit is not set in **dmFields**, **dmYResolution** is not used and **dmPrintQuality** retains the original meaning.

The **dmTTOption** member specifies how TrueType fonts should be printed. The member can be set to one of the following values.

Value	Meaning
DMTT_BITMAP (1)	Prints TrueType fonts as graphics.
DMTT_DOWNLOAD (2)	Downloads TrueType fonts as soft fonts.
DMTT_SUBDEV (3)	Substitutes device fonts for TrueType.

Note Before calling the the **CreateDC** or **CreateIC** functions, an application should call the **DeviceCapabilities** function with the DC_TRUETYPE index to retrieve the driver's TrueType capabilities. The application can use the value returned by the driver to set the **dmTTOption** member to the appropriate value.

The default action is to download TrueType as soft fonts for a printer that uses Printer Control Language (PCL); substitute device fonts for TrueType for Adobe PostScript® printers; and to print TrueType fonts as graphics for dot-matrix printers.

3.5.4 RESETDEVICE and STARTDOC Escapes

To support the **ResetDC** and **StartDoc** functions (new for Windows 3.1), printer drivers must process the RESETDEVICE and STARTDOC escapes in their **Control** functions.

Although the STARTDOC escape was available in Windows 3.0, the *lpInData* and *lpOutData* parameters have changed. Specifically, *lpInData* points to a null-terminated string specifying the name of the document, and *lpOutData* points to a DOCINFO structure specifying the output port or file, as well as the document name. The structure has the following form:

```
typedef struct {
    short   cbSize;
    LPSTR   lpszDocName;
    LPSTR   lpszOutput;         // output port name
} DOCINFO, FAR * LPDOCINFO;
```

The **lpszOutput** member is the name of the output file to use. If either *lpOutData* or **lpszOutput** is NULL, the output port given to the **CreateDC** function should be used.

The RESETDEVICE escape, corresponding to the new **ResetDC** function, allows the driver to move a printer's output state from an old physical device structure to a new one. This allows applications to change the printer setup, such as orientation, with creating a new print job. For more information about this escape, see the RESETDEVICE escape in the *Microsoft Windows Device Driver Adaptation Guide.*

3.5.5 Compatibility Tests for Printers

In addition to the Compatibility Test Suites for displays and networks Microsoft provided in earlier prereleases, we are now also providing a Test Suite for checking compatibility with printer drivers.

3.6 Keyboard Drivers

Windows 3.1 extends the set of virtual-key codes to support 122 key keyboards. The following section describes the extensions.

3.6.1 Virtual-Key Codes

The keyboard driver's hardware interrupt function translates OEM scan codes into virtual-key codes to provide Windows with keyboard input that is independent of the keyboard hardware.

The virtual-key code set consists of 256 byte values in the range 0 to 255. Most, but not all, of the values used by standard drivers are in the range 0 to 127. There is a loose distinction between "standard" keys, which do not vary much, and "OEM" keys, which do vary from keyboard to keyboard.

3.6.2 Required Virtual Keys

For full Windows functionality, a keyboard driver must provide the following virtual keys.

Virtual-key code	Description
VK_SHIFT	Left and right SHIFT keys.
VK_CONTROL	CTRL key.
VK_MENU	ALT key.
VK_A through VK_Z	Letter keys: A through Z.
VK_0 through VK_9	Number keys (not on numeric keypad): 0 through 9.
VK_UP	Up arrow key.
VK_DOWN	Down arrow key.
VK_LEFT	Left arrow key.
VK_RIGHT	Right arrow key.
VK_1 through VK_10	F1 through F10 keys.
VK_ESCAPE	ESC key.
VK_TAB	TAB key.
VK_SPACE	SPACE key.
VK_SNAPSHOT	PRINTSCREEN key.

Virtual-key code	Description
VK_BACK	BACKSPACE key.
VK_INSERT	INSERT key.
VK_DELETE	DELETE key.
VK_HOME	HOME key.
VK_END	END key.
VK_PRIOR	PAGE UP key.
VK_NEXT	PAGE DOWN key.

Shift keys must be available in the combinations unshifted, SHIFT, CTRL, and CTRL+ALT.

Note that on most keyboards, the key codes VK_INSERT, VK_DELETE, VK_HOME, VK_END, VK_PRIOR, and VK_NEXT are generated on the numeric keypad *only* if NUMLOCK is off.

3.6.3 Optional Virtual Keys

If a keyboard has a numeric pad, the numeric keys are frequently used as cursor-control and editing keys if NUMLOCK is off. If NUMLOCK is on, the virtual-key codes VK_NUMPAD0 through VK_NUMPAD9 are used for the digits. Keyboards with a DELETE key that also generates the decimal point (period or comma) use VK_DELETE and VK_DECIMAL to distinguish between the two uses of the key.

Keyboards commonly contain various "lock" keys, such as VK_CAPITAL and VK_NUMLOCK. If a keyboard driver generates ANSI characters on the numeric key pad using ALT + numeric-pad keys, it must do this translation only if NUMLOCK is on. Also, care must be taken (on IBM-compatible keyboards) that the cursor and editing keys on extended keyboards do *not* produce this translation.

Other keys may vary from keyboard to keyboard. The following set of virtual-key codes is generally used for punctuation keys, accented letter keys, and dead keys in the main section of a keyboard:

```
VK_OEM_1 .. VK_OEM_8
VK_OEM_102
VK_OEM_PLUS, VK_OEM_MINUS, VK_OEM_COMMA, VK_OEM_PERIOD
```

If a keyboard has more than 16 function keys, the virtual-key codes in the range VK_F17 through VK_F24 should be used for the extra function keys.

The mouse button virtual-key codes (VK_LBUTTON, VK_RBUTTON, VK_MBUTTON, and VK_CANCEL) are generated internally by Windows and are never generated by keyboard or mouse drivers.

Keyboard drivers should *not* generate VK_EXECUTE or VK_SEPARATER.

3.6.4 Virtual Keys for 122-Key Keyboards

For keyboards with 122 keys, the mapping for the first 101 keys is the same described in previous sections.

The function keys F13 through F24 should use VK_F13 through VK_F24.

The remaining keys should be mapped using the following virtual-key codes:

VK_ATTN
VK_CRSEL
VK_EXSEL
VK_EREOF
VK_PLAY
VK_ZOOM
VK_NONAME
VK_PA1
VK_OEM_CLEAR

3.7 Network Drivers

The new features for the 3.1 version of the network driver interface consist primarily of:

- Saved network connections
- New capability bits
- New return values
- String handling
- Error message handling
- New print, connection, and dialog functions

- Default shell behavior
- Long filename interface
- Enhancements to the MS-Net version 3.1 driver

Drivers written to the Windows 3.0 level of functionality will be supplied with reasonable defaults for all 3.1 features. If a network does not require special support for any of the features, the network driver may be written to the 3.0 specification.

3.7.1 Saved Network Connections

Saved network connections are connections which are always automatically connected when Windows starts. If the connection cannot be made, the drive is still treated as connected, as if it were in the error state. The File Manager will reconnect the drive if the user selects it. When the user connects to a drive or printer, that connection is remembered as a permanent connection. When the user disconnects a device through the disconnect dialog box, it is removed from the list of permanently connected devices.

Although error state and permanent devices can be reconnected from the File Manager, the drives may not (generally will not) be valid for other Windows applications. If Windows was unable to restore a permanent connection when starting, the user will be warned.

Network drivers which implement these functions will return a specification version of 3.10 (0x030A) from when the WNNC_SPEC_VERSION value is specified with the **WNetGetCaps** function.

3.7.2 New Capability Bits

The **WNetGetCaps** function now returns several new capability bits.

The WNNC_PRINTING value now includes WNNC_PRT_WriteJob (0x8000). This bit indicates support of the **WNetWriteJob** function.

The WNNC_CONNECTION value now includes WNNC_CON_RestoreConnection (0x0020). This bit indicates support of the **WNetRestoreConnection** function.

The WNNC_DEVICEMODE value has been renamed WNNC_DIALOG. The value includes several values for dialog functions, including a value for the **WNetBrowseDialog** function. Although a bit is reserved for **WNetBrowseDialog**

under WNNC_DIALOG, the WNNC_CONN_BrowseDialog bit (returned using WNNC_CONNECTION) must also be supported for compatibility; they mean the same thing and must both be set or clear.

The new WNNC_ADMIN (0x0009) value has been added to indicate support of long filenames and administration functions. In Windows 3.1, these are limited to modifying directory icons, and getting notification of directory manipulations.

The new WNNC_PRINTMGREXT (0x000B) value has been added to indicate support for extended Print Manager capabilities.

3.7.3 New Return Values

The **WNetGetConnection** function now returns several new error values.

The WN_DEVICE_ERROR (0x0035) value specifies that a connection exists on the given device, but that the connection is in an error state and should be disconnected and reconnected before it is accessed again. Generally, File Manager will attempt to reconnect such devices by calling the **WNetRestoreConnection** function.

The WN_CONNECTION_CLOSED (0x0036) value indicates that a permanently maintained connection is defined for the device, but that the device is not currently connected to any network resource. This return value is defined for those network drivers that export the **WNetRestoreConnection** function. The only way a permanent connection can be closed is if the device could not be connected when starting up.

The WN_CONTINUE (0x000D) value is used by functions to indicate a situation distinct from both WN_CANCELLED and WN_SUCCESS where File Manager does not need to post a message to the user.

3.7.4 String Handling

All strings passed to the network driver may be assumed to be ANSI strings. Similarly, all strings returned by the network driver should be ANSI strings. This is not a new feature.

3.7.5 Handling Error Messages

If appropriate, File Manager will post error messages whenever an error is reported by a network driver function. If the error value is WN_NET_ERROR, the **WNetGetError** and **WNetGetErrorText** functions will be called to allow the network driver to return a network specific error. If these two functions fail or are not implemented, a generic message will be used instead. This behavior will be standardized across all File Manager components. Some functions, such as connection dialogs, are allowed to post their own messages as part of processing the dialog box. File Manager never posts error messages if the return value is WN_SUCCESS, WN_CANCEL, or WN_CONTINUE unless otherwise noted.

3.7.6 New Print, Connection, and Dialog Functions

Windows 3.1 has the following new functions.

Function	Description
WNetWriteJob	Writes to an open print job.
WNetRestoreConnection	Restores a connection to a network device.
WNetConnectDialog	Displays a File Manager connection dialog box.
WNetDisconnectDialog	Displays a File Manager disconnection dialog box.
WNetConnectionDialog	Displays a connection dialog box.
WNetViewQueueDialog	Display a queue viewing dialog box.
WNetGetPropertyText	Adds buttons to a property dialog box.
WNetPropertyDialog	Supports added buttons in the property dialog.
WNetGetDirectoryType	Retrieves directory type.
WNetDirectoryNotify	Notifies driver of operations on directories.

For network drivers that provide user functions, the following ordinals are reserved the user functions:

WNNC_ORD_UserStart 500
WNNC_ORD_UserEnd 599

File Manager is guaranteed never to use ordinals in this range.

3.7.7 Default Shell Behavior

If a network driver does not support the new function calls (for example, it is a Windows 3.0 network driver), Windows and File Manager simulate a new method of supporting permanent connections.

The default connection dialog box includes a check box for making a connection permanent. If the user makes a permanent connection, the device will be added to a WIN.INI section called [Network]. The tag will be the local device and the value will be the network resource. For security, passwords will not be saved. The following example illustrates the new section:

```
[Network]
S:=\\USER2\SHELL
T:=\\TOOLSVR\DOSENV
U:=\\PYREX\USER
LPT1=\\WINDEV\PRINTER
```

When a device is disconnected, it is removed from the list of permanent devices. File Manager simulates the **WNetGetConnection** behavior of returning a connection with a special status when it is listed in WIN.INI but is not connected. The default connection dialog boxes allow the user to "reconnect" a disconnected device which has a saved connection. The dialog boxes also support disconnecting and reconnecting a device in an error state.

When starting, Windows enumerates devices saved in the [Network] section and attempts to redirect them using the **WNetAddConnection** function. If a device is already redirected, the redirection will not be overridden. If **WNetAddConnection** returns WN_BAD_PASSWORD, the user will be prompted to supply the password.

Network drivers which require their own connection dialog boxes or startup and restore functions should attempt to duplicate the functionality of the default behavior as closely as possible in order to provide consistency. To maintain this consistency, network drivers should use the [Network] section. The tags "A:" through "Z:" and all common MS-DOS device names ("LPT1", "COM1", "PRN", and so on) are reserved, including those not currently supported for redirection. The network driver may define additional tags of its own.

3.7.8 Long Filename Interface

The Long Filename interface enables access to files with names that do not fit the MS-DOS standard filenaming convention on network drives. The following list outlines these functions, and provides a brief description of each.

Function	Description
LFNFindFirst	Searches for the first file with a matching name.
LFNFindNext	Searches for the next file with a matching name.
LFNFindClose	Ends the search for matching files.
LFNGetAttributes	Retrieves file attributes.
LFNSetAttributes	Sets file attributes.
LFNCopy	Copies a file.
LFNMove	Moves a file.
LFNDelete	Deletes a file.
LFNMKDir	Creates a directory.
LFNRMDir	Removes a directory.
LFNGetVolumeLabel	Retrieves the volume label.
LFNSetVolumeLabel	Sets the volume label.
LFNParse	Parses paths.
LFNVolumeType	Specifies volume type.

Typically, these functions are called only when a network drive with long filenames is detected. In other words, they do not need to support local drives. The exception is the **LFNCopy** function which must be able to copy to or from any type of volume.

The return value is typically an MS-DOS error function (as returned from MS-DOS functions when the carry flag is set, or from Interrupt 21h Function 59h, Get Extended Error Information). If a function returns the special error code 0xFFFF, then the network error message functions will be called to retrieve error message text.

The maximum length of any long filename will be assumed to be 260 characters, including the terminator.

File-attribute values are identical to MS-DOS file-attribute fields.

3.7.9 Enhancements to the MS-Net Driver Version 3.1

The MS-Net driver 3.1 implements the error state return value from the **WNetGetConnection** function. All connection dialog boxes are the default File Manager dialog boxes. In addition, a WIN.INI switch is added to [Network] to disable the spooler queue viewing functions. This will allow better use of the MS-Net driver with "somewhat compatible" networks. By disabling the spooler-queue query functions, the MS-Net driver may run without assuming NetBIOS as the transport or SMB as the file-sharing protocol. It will also allow the use of the MS-Net driver with servers that do not correctly implement the **splretq** SMB command, such as early versions of Xenix®-Net.

If possible through the KERNEL module, the MS-Net driver should implement checking of open files when a device is disconnected.

3.8 Sample Driver: MOUSE.DRV

The source code for MOUSE.DRV included on the device driver disk is intended as a sample of a driver for Microsoft compatible mouse products. It is not equivalent to the retail version of the driver found on the Windows 3.1 disks.

This sample driver includes minor updates since Windows 3.0, and demonstrates compatibility with tablet drivers for pen-based systems.

3.9 New Features for Non-Windows Applications

Windows 3.1 provides performance enhancements, font selection, and mouse support for non-Windows applications running in a window. The table below describes these features, and indicates the locations of documentation and sample source code to assist with the addition of these features to third party drivers.

Feature	Description
Graphics applications in a window	The VGA VDD now supports execution of VGA graphics-mode applications in a window.
Smooth scrolling	The VDD has been improved to provide smoother scrolling of text based non-Windows applications.

Feature	Description
Improved color matching	The interface between the VDD and 386 enhanced-mode grabber has been changed to provide better representation of non-Windows application colors when running in a window.
Font selection	Font size can now be changed for text mode non-Windows applications running in a window. Changes are required in the 386 enhanced-mode grabber.
Mouse support	The interface between the virtual-mouse device (VMD), VDD, 386 enhanced-mode grabber, and the MS-DOS mouse driver has been enhanced to provide support for mouse operations in non-Windows applications.

3.10 Version Stamping

Windows 3.1 has version control of software components built into it. It is now a requirement for all Windows software components, including device drivers, to include version information. The File Installation Library includes functions that determine where a file should be installed, identifies conflicts with currently installed files, and performs the installation process. These functions enable installation programs to avoid the following problems:

- Installing older versions of drivers over newer versions
- Installing multiple copies of a driver in different directories
- Copying drivers to network directories being shared by multiple users

A developer can create a version resource that indicates a Windows file's purpose, author, version number, and so on. The File Installation Library also includes functions that enable applications to query this information and present it to the user in a clear format. For example, the Microsoft Diagnostics utility (MSD) uses the File Installation Library to provide a convenient method for product support technicians to identify files that are out of date, non-standard or otherwise mismatched.

The File Installation Library is available for Windows and non-Windows applications, and can be used by third-party software. For more information, see the Microsoft Windows 3.1 Software Development Kit documentation.

Utilities and Tools

Appendix A

This appendix lists and briefly describes some of the tools provided with the Microsoft Windows Device Driver Kit (DDK). These tools are necessary to prepare and build the sample device drivers and virtual devices.

File	Description
ADDHDR.EXE	Special Add Header utility for 386 virtual devices. This file modifies the headers of virtual devices. It takes only one parameter, the name of the virtual device file, and should be run as a standard part of every build procedure.
CVTHPPFM.EXE	Tool for converting from 3.1 PFM format to 3.0 PFM format.
IMPLIB.EXE	Tool for creating import libraries. For those developers who are still using the Microsoft C Compiler version 5.1, you should install the IMPLIB program provided with the compiler. But, because IMPLIB is *not* installed automatically, you need to do this manually.
LINK386.EXE	Link386 for 386 virtual devices. A version of the Microsoft Segmented-Executable Linker that builds virtual devices. This version has been modified to understand 32-bit offsets.
LINK4.EXE	A special version of the C Linker that creates font-resource files. The DDK includes this program to support the creation of font resources only. For other executable files, we recommend you use the linker provided with your compiler or assembler, or LINK386.EXE.
MAPSYM32.EXE	Special MAPSYM for 386 virtual devices. A 32-bit version of the standard MAPSYM utility that converts map files produced by the linker into symbol files used by some debuggers. This version has been modified to understand 32-bit offsets.
MASM.EXE	MASM 5.10A for building device drivers.

File	Description
MASM5.EXE	MASM 5.10B (flat model) for 386 virtual devices. A version of the Microsoft Macro Assembler, version 5.10B, that supports the 32-bit address space required for virtual devices. It has been named MASM5.EXE so it will not conflict with the standard MASM executables.
MKPRN.EXE	PostScript resource tool.
PFMEDIT.EXE	PFM Editor for PFM and PCM fonts.
SLIBCEW.LIB	The Windows version of the C version 6.0 runtime libraries.
TREEEX2.BAT	Used with TREEEXP.BAT to decompress DDK source files.
TREEEX3.BAT	Used with TREEEXP.BAT to decompress DDK source files.
TREEEXP.BAT	Used to decompress DDK source files.
WALK.EXE	Used by TREEXP.BAT to search for and decompress source files in directories.

Windows 3.1 Driver Filename Policy

Appendix B

Microsoft now requires that all device drivers, virtual devices, and related components have their filenames changed from the filenames used with the sample driver sources. In addition, for some drivers the driver module name must be changed, and for certain virtual devices the identification number must be changed.

This requirement eliminates the problem of an externally developed driver accidentally overwriting or conflicting with a standard Windows driver or virtual device. In addition, by having unique filenames for distinct drivers or virtual devices, our mutual product support personnel will be able to more easily determine when an externally developed driver has been installed.

B.1 Device-Driver Naming Conventions

The following sections discuss the filenaming policy.

B.1.1 Fixed Module Names

The following drivers have fixed module names that you must use for these types of hardware devices in Windows 3.1. If you create a new driver for one of these hardware devices based on Microsoft sample sources, you must use the corresponding module name. In addition you must use a unique driver filename.

Component	Module name
Keyboard	KEYBOARD
Display	DISPLAY
Mouse	MOUSE

Component	Module name
Communications	COMM
Sound	SOUND
386 Grabbers	GRABBER

B.1.2 No Module Names

The following display driver components do not contain module names:

- 286 grabbers
- Logo code
- Logo data

If you derive a new grabber or logo file based upon the samples provided, you must change the filename to a unique name.

B.1.3 Unique Module and Driver Filenames

For the following types of hardware you should provide a unique module name and filename:

- Installable drivers
- Printer drivers
- Network drivers

B.2 Distributing Display Driver Components

Microsoft encourages licensed DDK users to ship our fonts, grabbers, virtual display devices, logo files, and so on, but not change the filenames on their disks. The important rule is, if you *derive* a file from one of ours, then you must change its name. If it is identical, *don't* change it.

B.3 Virtual Devices

For virtual devices that replace standard devices included with Windows, you must use the same virtual device ID numbers as the devices they are replacing. For virtual devices which supplement, rather than replace, the devices in the standard Windows package, you may need to obtain a virtual device ID number from Microsoft DDK Product Support Services (PSS). Virtual device ID numbers are assigned by the DDK PSS. For all virtual devices created from the samples Microsoft provides, you must change the filename. For more information on contacting PSS, see Chapter 2, "Producing Device Drivers and Virtual Devices."

B.4 Version Stamping

You need to modify the version stamp to include correct information about your company and the specific file. For more information about File Version Stamping, see the Microsoft Windows 3.1 Software Development Kit documentation.

B.5 Choosing a Unique Filename

To increase the chance of picking a unique filename, Microsoft recommends that each Independent Hardware Vendor (IHV) use their company name in the name of the file.

B.6 Windows 3.1 Driver Filenames and Related Components

The following sections list the filenames used for drivers and virtual devices by Windows 3.1.

Display Drivers

8514.DRV
CPQAVGA.DRV
EGA.DRV
EGAHIBW.DRV
EGAMONO.DRV
HERCULES.DRV
PLASMA.DRV
SUPERVGA.DRV
V7VGA.DRV
VGA.DRV

	VGA.DRV
	VGAMONO.DRV
	VGAMONO.DRV
Standard-Mode Grabber	CGA.2GR
	CPQAVGA.2GR
	EGACOLOR.2GR
	EGAMONO.2GR
	HERCULES.2GR
	VGACOLOR.2GR
	VGAMONO.2GR
Logo Code	CGALOGO.LGO
	EGALOGO.LGO
	EGAMONO.LGO
	HERCLOGO.LGO
	VGALOGO.LGO
Virtual-Display Devices	V7VDD.386
	VDD8514.386
	VDDAVGA.386
	VDDCGA.386
	VDDEGA.386
	VDDHERC.386
	VDDVGA.386
	VDDVGA30.386
386 Enhanced-Mode Grabber	EGA.3GR
	HERC.3GR
	PLASMA.3GR
	V7VGA.3GR
	VGA.3GR
	VGA30.3GR
	VGADIB.3GR
Logo Data	CGALOGO.RLE
	EGALOGO.RLE
	HERCLOGO.RLE
	VGALOGO.RLE

Fonts for 386 Enhanced-Mode Grabbers

CGA40850.FON
CGA40WOA.FON
CGA80850.FON
CGA80WOA.FON
EGA40850.FON
EGA40WOA.FON
EGA80850.FON
EGA80WOA.FON
HERC850.FON
HERCWOA.FON

Keyboard Drivers

KEYBOARD.DRV

Mouse Drivers

MOUSE.DRV

Network Drivers

MSNET.DRV

System, Fixed, and OEM Fonts

Filename	Description
8514FIX.FON	8514/a (1024x768) resolution Fixed System font.
8514OEM.FON	8514/a (1024x768) resolution Terminal font (USA/Europe).
8514SYS.FON	8514/a (1024x768) resolution System font.
EGAFIX.FON	EGA (640x350) resolution Fixed System font.
EGAOEM.FON	EGA (640x350) resolution Terminal font (USA/Europe).
EGASYS.FON	EGA (640x350) resolution System font.
VGAFIX.FON	VGA (640x480) resolution Fixed System font.
VGAOEM.FON	VGA (640x480) resolution Terminal font (USA/Europe).
VGASYS.FON	VGA (640x480) resolution System font.

Miscellaneous Drivers

ADLIB.DRV
MMSOUND.DRV
SNDBLST2.DRV

Communication Drivers

COMM.DRV

Printer Drivers and Related Components

Filename	Printer
DMCOLOR.DLL	Color Library
EPSON24.DRV	Epson 24-pin printers
EPSON9.DRV	Epson 9-pin printers
FINSTALL.DLL	HP Font Installer
HPPCL.DRV	HP PCL 4 printers
HPPCL5A.DRV	HP PCL 5 printers
PAINTJET.DRV	HP PaintJet
PROPRN24.DRV	IBM Proprinter 24-pin printers
PSCRIPT.DRV	Adobe PostScript printer driver
UNIDRV.DLL	Microsoft Windows Universal Printer Driver

Virtual Devices

Filename	Device
BIOSXLAT.386	Virtual BIOS device
COMBUFF.386	Virtual COMM buffer device
EBIOS.386	Virtual EBIOS device
INT13.386	INT13 virtual device
LDOSNET.386	Local MS-DOS network device
PAGEFILE.386	Virtual page file device
PAGESWAP.386	Virtual paging device
VADLIBD.386	Virtual device for Ad Lib
VCD.386	Virtual COMM device
VDDCGA.386	Virtual CGA display device
VDDEGA.386	Virtual EGA display device
VDDHERC.386	Virtual Hercules display device
VDDV7VGA.386	Virtual Video Seven display device
VDDVGA.386	Virtual VGA display device
VDMAD.386	Virtual DMA device
VFD.386	Virtual floppy device
VKD.386	Virtual keyboard device
VMD.386	Virtual mouse device
VNETBIOS.386	Virtual NETBIOS device
VPD.386	Virtual printer device
VSBD.386	Virtual device for Creative Labs, Inc. device
WDCTRL.386	386 enhanced-mode block device

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®

0392 Part No. 29131