Getting Started

Microsoft® MODULAR WINDOWS™

SOFTWARE DEVELOPMENT KIT

Getting Started

Microsoft® Modular Windows™ Software Development Kit

Microsoft Corporation

Document No. MM37742-1192
Printed in the United States of America.

Contents

Introduction

The Microsoft® Modular Windows™ Software Development Kit (SDK) is a set of libraries, header files, tools, books, online Help, and sample source programs designed to help you create applications for TV-based multimedia players using the Microsoft Modular Windows Operating System.

The Modular Windows Software Development Kit is intended primarily for programmers familiar with the fundamentals of Windows programming (including C- and assembly-language programming).

This manual introduces the SDK and covers the following topics:

- Installing the SDK software
- Preparing your computer for developing applications for Modular Windows
- Reviewing the differences between Windows and Modular Windows
- Building and debugging applications that run under Modular Windows
- Getting support

For important information you'll need before you install the SDK, read Chapter 1, "Installing the Modular Windows Software Development Kit."

Getting your First Look at Microsoft Modular Windows

The Modular Windows SDK disc is designed to be run on a TV-based player. To get your first look at the Microsoft Modular Windows Operating System, put the SDK disc in the CD-ROM drive of the player and close the drive door. The player will launch Modular Windows and run the WinShell sample application. WinShell lets you run other sample applications located in the COOLDEMO directory on the SDK disc.

To exit WinShell, press the F7 key on the keyboard to bring up a toolbar and then choose the Exit button from the toolbar.

Redistributable Run-Time Modules

Some Microsoft Modular Windows features are included in dynamic-link libraries (DLLs) and can be used with Windows 3.1 as well as with Modular Windows. If you use these features in your applications for Windows 3.1, you must redistribute the DLLs and related support files. The Modular Windows SDK includes the following redistributable files:

File	Description
DISPDIB.DLL	DIB-display library
HC.DLL	Hand control library
HCSTUB.DLL	Hand control library for systems with no hand control
HCVIS.DLL	Hand control library for Tandy hand control
LRHELV.FON	Font file for low-resolution display driver
MCMAN.EXE	Memory-cartridge management utility (for VIS players only)
VISMOUSE.COM	Mouse driver for applications for MS-DOS (for VIS players only)
TVUI.DLL	TV user-interface control library
TVVGA.FON	Font file for high-resolution display driver

The GDI.EXE and USER.EXE files are **not** redistributable. For more information about redistributable run-time modules, read Section 5 in the separate Microsoft Modular Windows SDK license agreement.

The multimedia video and audio materials included in the Modular Windows SDK are all copyrighted, and may not be reproduced, distributed, or publicly performed. For a listing of the copyright owners to contact regarding permission and use rights for these materials, see the copyright page of this manual. Note that the Modular Windows SDK does not contain any sample video or audio materials that you may redistribute as mentioned in Section 3 of the Microsoft Modular Windows SDK license agreement.

Books in the Modular Windows Software Development Kit

The Modular Windows SDK includes the following books that contain information on designing and programming applications for Microsoft Modular Windows:

- *Getting Started* introduces the Modular Windows SDK, explains how to install the SDK, and describes how to create a basic application for Modular Windows. You should read this book first to learn the differences between creating applications for Microsoft Windows 3.1 and creating applications for Modular Windows.
- *Programmer's Reference* describes Modular Windows support of the Windows 3.1 API and the Windows extension libraries. It also includes an overview and details on the new Modular Windows API.
- *Design Guide* provides guidelines for developing user interfaces for applications that run on the Modular Windows Operating System. Following these guidelines will help you enjoy the benefits of consistency between your application's interface and other applications for Modular Windows.

Online Information

Much of the information in the *Programmer's Reference* is also available online in Microsoft Windows Help (WINHELP.EXE) format in a file named MWREF.HLP. After installing the SDK, you can start MWREF.HLP by choosing the MWREF.HLP icon from the Modular Windows SDK group, or by using the Open command on the Windows Help File menu to open MWREF.HLP.

Obtaining Support

The Windows Extensions forum on CompuServe® provides online technical information for developers using the Modular Windows SDK. Using this forum, you can exchange messages with Microsoft professionals and experienced Microsoft users. You can also download free software, such as drivers, patches, tools and add-ons provided by Microsoft and CompuServe members.

To connect to the Windows Extensions forum, type **GO WINEXT** at the Compuserve "!" prompt. For information about establishing a CompuServe account, call (800) 848-8199, 8:00 A.M. to 10:00 P.M. EST. Ask for operator 230 and receive a $15 connect-time usage credit.

CHAPTER 1

Installing the Modular Windows Software Development Kit

This chapter explains how to install the Microsoft Modular Windows Software Development Kit (SDK). You should read this chapter carefully before installing the SDK.

Hardware and Software Requirements

The Modular Windows SDK lets you develop applications for the Microsoft Modular Windows Operating System. This section lists the recommended hardware and software configurations for using the Modular Windows SDK.

You can install and use the Modular Windows SDK on any computer that meets the minimum qualifications for Microsoft Windows version 3.1. However, for application development, the following hardware is recommended:

- Windows-compatible PC using an 80386 or later microprocessor
- At least 4 MB of RAM
- Hard disk with at least 20 MB available space
- A CD-ROM drive that meets the multimedia PC (MPC) standard
- A sound board that meets the multimedia PC (MPC) standard
- A VGA-NTSC display adapter and television that are compatible with Microsoft Windows
- A hand control that is compatible with Microsoft Modular Windows

In addition to a computer for application development, you'll need a TV-based player to test your application. You should thoroughly test your application on all players running the Modular Windows Operating System.

Before you install the SDK, be sure the following software is already installed on your development system:

- MS-DOS 3.22 or later
- Microsoft Windows version 3.1 Operating System
- Any compiler or assembler (and accompanying linker) that supports development of applications for Microsoft Windows 3.1

Before Installing the Software Development Kit

Because the Modular Windows SDK Setup program is an application for Windows, you must have Windows 3.1 installed on your development system before you install the Modular Windows SDK.

In addition, you should create a second installation of Windows for use with the Modular Windows SDK. This installation does not need to include any of the optional files in Windows, such as samples and help.

▶ **To create a second Windows installation:**

1. Run the Windows Setup program and choose the Custom Setup option.
2. The Windows Setup program displays a dialog box asking for the directory where you want to set up Windows. Change the path to set up Windows in a new directory called MODWIN.
3. The Windows Setup program displays a dialog box of Setup options. Select the Set Up Only Windows Components You Select option. Clear the options to set up printers and applications already on hard disk.
4. The Setup program displays a dialog box letting you select optional groups of files to install. Clear all of the optional groups.
5. After installing files, Windows Setup displays a dialog box letting you choose how to make modifications to AUTOEXEC.BAT and CONFIG.SYS files. Choose the option letting you make modifications later. Don't choose the option to let Setup make these modifications for you.

For more information about using this second installation of Windows, see "After Installing the Software Development Kit," later in this chapter.

Using the Setup Program

The Modular Windows SDK Setup program installs the Modular Windows Software Development Kit (SDK) on your development system. Setup copies the Modular Windows SDK files to your hard disk, adds the Modular Windows SDK group to your Program Manager window, and modifies the initialization files in your Windows directory. Optionally, Setup copies the sample program sources, online information files, and libraries and installs drivers for the Gravis PC GamePad™.

Important You must be running Windows from your normal Windows installation to install the Modular Windows SDK. Don't run the Modular Windows SDK Setup program while running Windows from your second Windows installation.

▶ **To install the Modular Windows SDK:**

1. Insert the CD-ROM disc into your CD-ROM drive.
2. You can run the Setup program from Program Manager or File Manager.

   To run Setup from Program Manager, choose Run from the File menu and type the path to the Setup program. For example, if you are installing the SDK from your CD-ROM drive (drive E), type the following command line:

   ```
   e:\setup.exe
   ```

   To run Setup from File Manager, select your CD-ROM drive, then double-click SETUP.EXE in the root directory of the drive.
3. The Setup program displays a dialog box letting you choose the directory where you want to install the SDK. Install the SDK in the directory you used for the second Windows installation. This directory should be called MODWIN.
4. Setup displays a dialog box letting you select the SDK components you want to install. By default, Setup assumes you want to install the entire SDK. If you don't want to install a particular component, clear the check box for that component.

   If you won't be using a Gravis PC GamePad as a hand control, don't install this component of the SDK.

5. Setup displays a dialog box letting you choose the directory where you want to install the run-time files. The run-time files include dynamic-link libraries (DLLs) and other files required to run Modular Windows. Because these files must be available to both installations of Windows, Setup installs these files in a separate directory.

   You shouldn't need to change the default directory setting supplied by Setup.

6. To begin installing the SDK files, choose Continue. Setup copies the SDK files to your disk and then creates the Modular Windows Program Manager group and icons.

7. Depending on the video card and driver you're using, you might need to modify your WIN.INI file. To determine whether this is necessary, refer to the documentation included with your video card.

8. After installation is complete, exit Windows and run the NEW_VARS.BAT program from the MS-DOS® prompt. This batch file defines environment settings required to use the Modular Windows SDK.

   If you use a text editor to add these environment settings to your AUTOEXEC.BAT file, the PATH entries from NEW_VARS.BAT must precede the existing PATH entries.

Setting Up Hand-Control Drivers

Modular Windows supports two hand-control devices: the Gravis PC GamePad and the Tandy® hand-control. This section explains how to set up Windows to use these hand control devices.

Installing the Gravis PC GamePad

If you choose the Modular Windows SDK Setup option to install the Gravis PC GamePad component, Setup automatically makes changes to the SYSTEM.INI files in both Windows installations by adding the following lines to SYSTEM.INI:

```
[boot]
drivers=mmsystem.drv hc.drv

[drivers]
joystick=ibmjoy.drv
```

You don't need to make any additional changes to SYSTEM.INI to use the Gravis PC GamePad.

Installing the Tandy Hand Control

To use the Tandy hand control, you must install the Tandy hand-control driver.

▶ **To install the Tandy hand-control driver:**

1. Copy the HCVIS.DLL file from your \MODWIN\OEM\VIS directory to the directory containing the run-time files for Modular Windows (\MWDIST).
2. Rename HCVIS.DLL as HC.DLL.
3. Restart Windows.

Changing the Installed Hand Control

If you install the Gravis PC GamePad and later want to use either the Tandy hand control or no hand control, you must replace the Gravis PC GamePad driver with the appropriate driver.

▶ **To replace the Gravis PC GamePad driver:**

1. Remove HC.DRV from the drivers= line in the SYSTEM.INI file. The drivers= line should now read as follows:

   ```
   drivers=MMSYSTEM.DLL
   ```

2. Remove the joystick= line from SYSTEM.INI.
3. To install the Tandy hand-control driver, copy the HCVIS.DLL file from your \MODWIN\OEM\VIS directory to the directory containing the run-time files for Modular Windows (\MWDIST).

   If you don't want to use a hand control, rename the HCSTUB.DLL file in the \MWDIST directory to HC.DLL.

Choosing a Windows Installation

You can do most of your development, testing, and debugging running Windows from your normal Windows 3.1 installation. However, in the final stages of testing, you should test your application running Modular Windows from your second Windows installation.

To choose between the two installations of Windows, edit the PATH environment setting in your AUTOEXEC.BAT file to include the appropriate Windows directory. You can create one batch file to run Windows 3.1 and a second batch file to run Modular Windows.

The Modular Windows SDK includes both debugging and nondebugging versions of the GDI.EXE and USER.EXE modules. You can use the MWN2D.BAT and MWD2N.BAT batch files to switch between the debugging and nondebugging versions of Modular Windows.

Running the WinShell Application

While running Modular Windows, you won't be able to use Program Manager or other applications not designed to run under Modular Windows. The Modular Windows SDK Setup program installs a SYSTEM.INI file in the second Windows installation that runs a shell application named WinShell. The WinShell application lets you launch applications and use the Media Control Interface (MCI) to play multimedia data files. You can also start Windows directly into your application by editing the shell= line in the [boot] section of the SYSTEM.INI file. The source code for WinShell is provided as a sample application.

To exit WinShell, press the F7 key on the keyboard to display a toolbar and then choose the Exit button from the toolbar.

CHAPTER 2

What's New for Modular Windows?

The Microsoft Modular Windows Operating System is based on a subset of the Microsoft Windows 3.1 Operating System with extensions to support TV-based multimedia players. This chapter provides an overview of the differences between the Windows 3.1 SDK and the Modular Windows SDK.

Changes from Microsoft Windows 3.1

The Modular Windows SDK offers the following changes from the Microsoft Windows 3.1 SDK:

- Reduced support for the Windows 3.1 application programming interface (API)
- Reduced support for Windows 3.1 extension libraries
- New user-interface controls
- New support for hand-control input devices

The following sections describe these changes in more detail.

Support for Microsoft Windows 3.1 API and Extension Libraries

Most of the Microsoft Windows 3.1 functions are fully supported in Modular Windows. Others are partially supported; for example, the WS_MAXIMIZEBOX window style is not available in the **CreateWindow** function. Some functions from Windows 3.1 are not supported in Modular Windows. For example, functions that result in a write operation to a disk are not supported on TV-based players that don't support writeable file devices.

Some of the Windows 3.1 extension libraries are directly supported in Modular Windows. Others are supported with some restrictions for platforms without writeable file devices. Some libraries, such as the common dialog-box library (COMMDLG.DLL), are not supported in Modular Windows.

See Chapter 4, "Core API and Extension Libraries Support," in the *Modular Windows Programmer's Reference* for detailed information about support for the Windows 3.1 API and extension libraries.

New Support for TV-Based Players

Modular Windows provides new support for TV-based multimedia players. The following sections discuss the new elements of support for players.

New User-Interface Controls

Modular Windows provides new user-interface controls designed to be easier to use and display well on televisions. Many of the new user-interface controls can be customized with application-supplied bitmaps and colors. Most of the familiar controls from Microsoft Windows 3.1, such as buttons, scroll bars, and list boxes have equivalent controls in Modular Windows.

Modular Windows does not support menus or window borders for sizing and moving windows. For complete information about Modular Windows user-interface controls, see Chapter 1, "User-Interface Controls," in the *Modular Windows Programmer's Reference.*

New Support for Hand-Control Input Devices

The primary input device for TV-based players is the infrared or wired hand control. Modular Windows provides full support for the hand control. The user interface is designed to operate with the hand control, as well as with a mouse or keyboard. Applications can also receive hand-control events in window procedures in much the same way as they receive mouse and key events.

For complete information about Modular Windows support for the hand control, see Chapter 2, "Hand-Control Services," in the *Modular Windows Programmer's Reference.*

New Tools for Programming and Debugging Applications

The Modular Windows SDK includes the following new tools for programming and debugging applications for Microsoft Modular Windows:

- Microsoft Transport Layer TSR (TLTSR.EXE), a tool that provides serial communication between a PC and a TV-based player.
- Microsoft Redirected File Server (RFSERVER.EXE), a tool that works in conjunction with the Transport Layer TSR to provide redirected file services on a TV-based player. Redirected file services let you use files residing on a host PC's hard disk to replace or supplement files on the player's CD-ROM disc.
- Microsoft Modular Windows 80286 Debugger (WDEB286.EXE), a debugger for applications running on TV-based players using the 80286 microprocessor.
- NoEcho (NOECHO.EXE), a tool that works in conjunction with Redirected File Server and Transport Layer TSR to display debugging messages generated by an application running on a TV-based player.
- Microsoft Modular Windows Heap Walker (MODWHEAP.EXE), a tool that lets you examine the local and global heaps used by applications and dynamic-link libraries (DLLs) running under Microsoft Modular Windows.
- Microsoft MS-DOS Monitor (DOSMON.COM), a tool that reports all calls to unsupported MS-DOS functions.
- Microsoft Color Table Converter (CTLCONV.EXE), a tool that converts bitmap and icon color tables.
- Conver24 (CONVER24.EXE), an application for Windows that converts images to new file formats supported by Modular Windows.

For details about using these tools, see Chapter 9, "Tools," in the *Modular Windows Programmer's Reference*.

CHAPTER 3

Creating Applications for Modular Windows

This chapter explains how you create applications for Modular Windows and how to run your applications on TV-based multimedia players. It also provides guidelines for debugging and testing your applications.

Writing Applications for Microsoft Modular Windows

The process of creating applications for Microsoft Modular Windows is similar to the process of creating applications for Microsoft Windows 3.1—if you have experience programming applications for Windows 3.1, you'll find it easy to learn how to write applications for Modular Windows.

To create an application for Modular Windows, you use the same basic tools, header files, and libraries you use to create applications for Microsoft Windows 3.1, plus the new tools, header files, and libraries included in the Modular Windows SDK.

Note Some parts of the Windows 3.1 API have either changed or are not available in Modular Windows. See the *Modular Windows Programmer's Reference* for details on the changed and unavailable parts of the Microsoft Windows 3.1 API.

New Header Files

The Modular Windows SDK includes the following new header files. These header files depend on declarations made in the WINDOWS.H header file. You must first include WINDOWS.H if you include any of the following header files:

- DISPDIB.H (for the DIB-display services)
- HC.H (for hand-control services)
- TVUI.H (for user-interface services)
- MC.H (for VIS™ memory-cartridge services)

In addition, the Modular Windows SDK includes MODW_API.H, a version of WINDOWS.H with unsupported functions, messages, and window styles removed. You can use this header file as a replacement for WINDOWS.H to ensure your application does not use any part of the Microsoft Windows 3.1 API not supported by Modular Windows.

New Libraries

The Modular Windows SDK also includes the following import libraries. If your application uses any of the new Modular Windows functions, you must link to the corresponding import library.

- DISPDIB.LIB (for the DIB-display services)
- HC.LIB (for hand-control services)
- TVUI.LIB (for user-interface services)
- MC.LIB (for VIS memory-cartridge services)

Running Applications on a TV-Based Player

Applications for Modular Windows can be run under Microsoft Windows 3.1 on a development system by launching the application from a shell program, such as Program Manager. However, because Modular Windows does not have a shell program, the launch process for applications run on a TV-based player is different. The following is the launch process for Modular Windows on a TV-based player:

1. When a disc is inserted into the player, the player executes a launch file called AUTOEXEC. The launch file starts Modular Windows. AUTOEXEC must be in the root directory of the CD-ROM disc.
2. During startup, Modular Windows reads the SYSTEM.INI file from the CD-ROM disc and runs the application specified by the "shell" keyname in the [boot] section of SYSTEM.INI.

Note VIS players require that application discs contain an additional launch file called CONTROL.TAT. Information about creating the CONTROL.TAT file is available from the VIS manufacturer. See Appendix B, "VIS Programming Notes," in the *Modular Windows Programmer's Reference* for details about obtaining additional information and tools for VIS development.

The following sections provide details about the launch file, the SYSTEM.INI file, and organizing files on the CD-ROM disc.

Creating the AUTOEXEC Launch File

The AUTOEXEC launch file starts Modular Windows and sets environment variables. It's similar to a batch file, but more restrictive in its format and capabilities. AUTOEXEC must be located in the root directory of the CD-ROM disc.

Important Although the version of Modular Windows in VIS players doesn't require that application discs contain an AUTOEXEC launch file, future versions of Modular Windows in other players will require this file. To ensure your disc is compatible with players other than VIS, you must include an AUTOEXEC launch file.

AUTOEXEC Launch-File Format

The AUTOEXEC launch file is an ASCII text file with the following format restrictions:

- Filename extensions must be specified for executable files.
- All paths must be fully qualified—relative paths are not allowed.
- Lines are limited to 128 characters, including carriage-return and line-feed characters.
- Blank lines and comments are not allowed.

Starting Microsoft Modular Windows

The only command you must include in your launch file is the command to launch Modular Windows, the **modwin** command. The **modwin** command has the same purpose as the **win** command used to start Windows 3.1 on a PC.

The following line shows **modwin** command-line syntax:

modwin *windir* [*environment1*, *environment2*, ...]

The *windir* argument specifies the Windows directory to be used during the session of Modular Windows. The Windows directory contains extension libraries, supplemental device drivers and .INI files. This directory must contain a SYSTEM.INI file specifying the name of the application for Modular Windows to run.

Note The path specified by the *windir* argument must not end with a backslash (\) character. To specify the root directory, include only the drive name.

The *environment* arguments specify optional environment variables that the application can use. The format for specifying environment variables is as follows:

name=setting

White space (not a semicolon) is the delimiter for environment variables—don't use any space before or after the equal sign.

Examples

The following line is a launch file that starts Modular Windows. The command argument specifies the directory containing the SYSTEM.INI file used to configure Modular Windows and specify the name of the application to run, as shown in the following command:

```
modwin a:\windows
```

The following command launches Modular Windows using the SYSTEM.INI file in the root directory on the disc and sets two environment variables, PATH and HELP:

```
modwin a: path=a:\sounds a:\images help=a:\help
```

The Microsoft Modular Windows File-Search Order

The only environment setting used by Modular Windows is the PATH environment setting—other environment settings are used only by applications. The following is the search order Modular Windows uses to locate executable and dynamic-link library (DLL) files:

1. The directory the application resides in
2. The current directory (unless changed by the application, the current directory is the root directory on the CD-ROM drive)
3. The Windows directory (specified at launch time of Modular Windows)
4. A subdirectory named SYSTEM (if it exists) in the Windows directory
5. The directories in the PATH environment setting, in the order specified

Running MS-DOS Applications

In addition to launching Modular Windows, you can also run MS-DOS applications, including terminate-and-stay-resident (TSR) programs from the AUTOEXEC launch file. To run an MS-DOS application, specify the complete path of the executable file on a line before the line containing the **modwin** command. The filename must include a file extension (generally, .EXE or .COM).

Example

The following is an example of an AUTOEXEC launch file that runs an MS-DOS application before launching Modular Windows:

```
\init.exe
modwin a:
```

Creating the SYSTEM.INI File

The *windir* argument of the **modwin** command specifies the Windows directory used during the session of Modular Windows. The directory specified by the *windir* parameter must contain a SYSTEM.INI file. In addition to specifying system-configuration information, the SYSTEM.INI file specifies the name of the application for Modular Windows to run.

Specifying the Application to Run

Use the shell= setting in the [boot] section of SYSTEM.INI to specify the name of the application for Modular Windows to run on startup. For example, the following entry in SYSTEM.INI causes Modular Windows to run an application called YOURAPP.EXE:

```
[boot]
shell=yourapp.exe
.
.
.
```

Specifying System Configuration

In addition to specifying the application that you want to run, SYSTEM.INI also specifies system-configuration information used by Modular Windows such as fonts, device drivers, and MCI drivers. To launch Modular Windows, your SYSTEM.INI file must contain the following entries:

```
[boot]
shell=yourapp.exe
mouse.drv=mouse.drv
sound.drv=sound.drv
comm.drv=comm.drv
system.drv=system.drv
drivers=mmsystem.dll
oemfonts.fon=vgaoem.fon
fixedfon.fon=vgafix.fon
fonts.fon=vgasys.fon
display.drv=vga.drv
keyboard.drv=keyboard.drv

[drivers]
wave=vwavmidi.drv
midi=vwavmidi.drv
timer=timer.drv

[mci]
AVIVideo=mciavi.drv
CDAudio=mcicda.drv
Sequencer=mciseq.drv
WaveAudio=mciwave.drv
```

Don't use the SYSTEM.INI file to specify private application-configuration information—use the WIN.INI file for this purpose.

Adding Extension Libraries

You can add Windows extension libraries, such as the Data Decompression Library (LZEXPAND.DLL), to your application by including the DLL file in a subdirectory called SYSTEM in your Windows directory (the directory specified on the **modwin** command line). For example, if your Windows directory is \WINDOWS, put extension-library files in the \WINDOWS\SYSTEM directory. Modular Windows does not completely support all of the Windows 3.1 extension libraries. See Chapter 4, "Core API and Extension Libraries Support," in the *Modular Windows Programmer's Reference* for details on extension libraries supported by Modular Windows.

Changing Display Drivers and Fonts

The name of the display driver in the Modular Windows ROM is VGA.DRV (TVVGA.DRV is the corresponding driver for use on development systems with Tandy® TVVGA display interfaces). To use the Modular Windows display driver, you must include the following entry in your SYSTEM.INI file:

```
display.drv=vga.drv
```

This display driver is a dual-mode driver supporting both high-resolution (640-by-400) and low-resolution (320-by-200) modes. The default mode is high resolution. To use the low-resolution mode, add the following section to SYSTEM.INI:

```
[TVVGA]
resolution=320x200x8
```

If you use the low-resolution display mode, you can include the LRHELV.FON font file on your disc. Put this font file in the Windows directory or in a subdirectory of the Windows directory called SYSTEM. To install the font, you must include a WIN.INI file with the following entry:

```
[windows]
systemfont=lrhelv.fon
.
.
.
```

Creating the WIN.INI File

If your application uses the Media Control Interface (MCI), you might want to include a WIN.INI file with the following entries:

```
[mci extensions]
wav=waveaudio
mid=sequencer
rmi=sequencer
AVI=AVIVideo
```

The [mci extensions] entries allow MCI to automatically associate files with the appropriate MCI device driver.

The WIN.INI file should be located in the Windows directory (along with SYSTEM.INI). You can use the WIN.INI file to set configuration information for Modular Windows as well as initialization information for your application.

WIN.INI Caching Considerations

Modular Windows always caches the WIN.INI file, so it's a good place to put private initialization information. Because private initialization files are not guaranteed to be cached, you might notice reduced performance when you attempt to read from a private initialization file.

Note To ensure Modular Windows properly uses the initialization-file cache, don't specify paths when reading from WIN.INI or SYSTEM.INI. As long as these files are in the Windows directory, you can access them with the initialization-file functions by specifying only the filename of the initialization file.

CD-ROM Directory Structure

The organization of directories and files on your CD-ROM disc will affect the performance of your application. The CD-ROM device driver caches a path table and directory table to prevent rereading path and directory information from the disc each time a new file is opened. To keep the path and directory tables small enough to remain in the cache, consider the following guidelines when you create your CD-ROM directory structure:

- If your disc contains less than 50 files, put all files in the root directory.
- If your disc contains more than 50 files, put functional subgroups of files in the same directory. For example, if your application is an encyclopedia, put all files (hypertext, audio, images, etc.) related to a certain topic in the same directory.
- Limit the number of directories you use to 100.

The values given above are approximate values. The actual values depend on the number of characters in the filenames and directory names. Using shorter names, you can have more files and directories and still keep the path and directory tables small enough to remain in the CD-ROM device-driver cache.

The following is the formula for the maximum number of files that will fit in the directory table for a given average filename length:

Number of files = 2048 / (33 + length of average filename)

The following is the formula for the maximum number of directories that will fit in the path table for a given average directory-name length:

Number of directories = 2048 / (9 + length of average directory name)

The length of the average directory name does not include the full path.

Debugging and Testing Applications

The following is the suggested process for developing and debugging applications for Microsoft Modular Windows:

- Compile your application using the MODW_API.H header file to replace the WINDOWS.H header file. This will verify your application does not use parts of the Microsoft Windows 3.1 API not supported by Modular Windows.
- Develop your application under the debugging version of Microsoft Windows 3.1. The debugging version will detect many inadvertent mistakes before they turn into hard-to-find bugs. Test the application in standard mode as well as enhanced mode.
- Once the application is running, run it under the Modular Windows versions of the GDI.EXE and USER.EXE modules to verify the application doesn't make calls to unsupported parts of the Windows API. The versions of GDI.EXE and USER.EXE included in the Modular Windows SDK are functionally identical to the modules in the ROM.
- Next, begin final testing and fine-tuning of the application on a TV-based player. Test the application using different types of televisions to ensure your application appears as you designed it. The SDK includes debugging tools you can use to test and debug on a TV-based player.

For details about debugging using the tools included in the Modular Windows SDK, see Chapter 9, "Tools," in the *Modular Windows Programmer's Reference*. For details about using the debugging version of Windows 3.1, see the *Getting Started* manual in the Microsoft Windows 3.1 SDK.

Using the Modular Windows Versions of GDI.EXE and USER.EXE

The Modular Windows SDK includes nondebugging and debugging versions of the Windows core GDI.EXE and USER.EXE modules. You can use these modules to test your application thoroughly on a development system before running it on a TV-based player.

Because the nondebugging Modular Windows versions of GDI.EXE and USER.EXE are identical to the versions in the ROM, you won't be able to run Program Manager to launch your application. Instead, you can launch your application by setting the shell= entry in the [Boot] section of SYSTEM.INI.

The SDK Setup program creates two directories that contain the debugging and nondebugging versions of the core modules. Setup places the debugging versions in the DEBUG subdirectory and the nondebugging versions in the NODEBUG directory in the subdirectory where you installed the Modular Windows SDK.

Using File Redirection

Two tools included in the SDK, TLTSR.EXE and RFSERVER.EXE, provide file redirection during testing and debugging. File redirection lets you use files residing on a host PC's hard disk to replace or supplement files on the player's CD-ROM disc. File redirection can save time and expense during development by reducing the number of CD-ROM discs you make for testing. For details about using file redirection, see Chapter 9, "Tools," in the *Modular Windows Programmer's Reference*.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Microsoft®