Getting Started

Microsoft® Win32®

Software Development Kit

Legal Information

Introduction

The Microsoft® Win32® Software Development Kit (SDK) provides a complete development environment for the creation of Win32-based applications for any Microsoft operating system that supports Win32-based applications, including Win32s®, Windows® 95, and Windows NT™.

This document presents an overview of the SDK's contents and provides essential installation information. The topic Installing the Win32 SDK provides the minimum information required for installation of this development kit.

Package Contents

- One CD-ROM containing Windows 95
- One CD-ROM containing the Win32 SDK for Microsoft Windows (supports *x*86, MIPS, Alpha AXP, and PowerPC platforms)

Installing the Win32 SDK

This topic provides information on installing the Win32 Software Development Kit (SDK). It is assumed that you have already installed the Microsoft Windows NT version 3.51 operating system or the Microsoft Windows 95 operating system.

The Microsoft Developer Network includes both Windows NT Workstation version 3.51, and Windows 95 on CD-ROM only. To install Windows NT on *x*86 computers, use either WINNT.EXE or WINNT32.EXE. To install Windows 95, use SETUP.EXE, in the root. There is no support for floppy disk installation.

System Requirements

The Win32 SDK for Microsoft Windows requires the following minimum configuration:

- Windows NT 3.51 or Windows 95
- CD-ROM drive
- 16 MB RAM (recommended for development)
- 45 MB available hard-disk space for a minimal SDK installation; 160 MB for a full installation

The Win32 SDK is available only on compact disc. You must have a CD-ROM drive installed locally or accessible from the network. For a list of computer systems, CD-ROM drives, SCSI adapters, and other peripherals compatible with Windows NT 3.51, see the Hardware Compatibility List help file called MAY95HCL.HLP in the <*CD_DRIVE*>\DOC \WNTHCL directory.

Win32 Installation

There are two ways to install the Microsoft Win32 SDK: graphical and manual (text based). The graphical installation is recommended, and it runs on both Windows 95 and Windows NT. To use this method, you must be able to access the CD-ROM drive from your Windows installation either locally or over a network. Use the text-based installation from MS-DOS® only if you cannot access the CD-ROM from Windows NT.

Graphical Installation

1. From Windows NT or Windows 95, run the SETUPSDK.BAT file from the root directory of the CD-ROM.

   A graphical Setup program prompts you to install the tools, headers, libraries, and samples on your system.
2. When Setup has finished, each new command window in Windows NT has environment variables correctly set to build Win32-based applications. On both Windows NT and Windows 95 setup also creates a program in the Win32 SDK Tools group called "Set Win32 Environment." Launching this program provides a command prompt with the appropriate environment set. If you install the Win32 SDK using an Administrator account on Windows NT, all users of the system can access the Win32 SDK from a common Win32 SDK Program Group created during installation.

Manual Installation (Windows NT Only)

1. Boot MS-DOS.
2. Go to the **<***cd_drive:***>**\MSTOOLS directory of the compact disc.
3. Type **manual <*drive:*>** to install the Win32 SDK in the **<***drive:***>**\MSTOOLS directory on the hard drive.

   A batch file copies all the tools, headers, libraries, and samples to the hard drive.
4. Reboot Windows NT.
5. From the command prompt, type **setenv <*drive*>:\mstools** from your **<***drive***>:**\MSTOOLS directory on the hard drive. This sets the environment variables for development. Alternatively, you can choose the System icon from Control Panel and permanently set environment variables for all command sessions.

SDK Features

The Win32 SDK provides a number of features to help you develop stable and powerful applications. Most of these features are supported on both Windows 95 and Windows NT. However, some features that are specific to Windows 95 are not yet available on Windows NT. Such features and functions are marked as "New - Windows 95" in the documentation. Features include:

- An integrated environment for targeting Win32s, Windows 95, and Windows NT
- Remote Procedure Call Support (RPC)
- OLE
- Multimedia
- Win32s
- InstallSHIELD Setup Toolkit
- Microsoft Setup Toolkit
- OpenGL™ (Windows NT only)
- POSIX (Windows NT only)
- Software Compatibility Test (SCT) (Windows NT only)
- Shell Extensions (New - Windows 95)
- Pen Windows (New - Windows 95)
- Telephony API (TAPI) (New - Windows 95)
- Messaging API (MAPI) (New - Windows 95)

Integrated Environment

The Win32 SDK now includes full support for all Microsoft's 32-bit operating systems (OS). The goal of this kit was to make developing one 32-bit application to run in all 32-bit environments as simple as possible. To this end, all the headers, libraries, and samples that used to exist separately in the Win32 SDK for Windows NT and the Windows 95 SDK have been merged into one common set of files and directories. You can install the Win32 SDK on either Windows NT or Windows 95, and write one application that will run on both operating systems. You can even install the SDK on one OS and write an application that targets the other OS exclusively. This section describes how each of the major areas is organized, and what platform differences you can expect in each area.

At the Win32 level it is easy to write one application to run on both platforms. All the major functionality sets in GDI, Kernel, User, Multimedia, RPC, and OLE are supported on both Windows 95 and Windows NT. However, there are some differences between the two environments. Windows 95 supports new API sets such as TAPI, PEN, and ICM, which are not yet implemented on Windows NT. Windows NT supports advanced features such as security, Unicode, and services, which are not implemented on Windows 95. This release of the Win32 SDK is geared toward making it easy to write one application to run in both environments, while still making these differences clear.

Tools

Most of the tools used to create 32-bit applications are common across both operating systems. However, due to certain features of the OS (such as advanced profiling functionality on Windows NT, and PEN support on Windows 95), some tools can only run on one OS or the other. These tools exist in \WINNT and \WIN95 subdirectories on the CD. When you install the SDK, Setup detects which OS you are running on and installs only the tools that work on that OS.

Headers/Libraries

All the headers and libraries for the Win32 API are now common between Windows NT and Windows 95. You no longer has to use two different sets of headers to build an application. If a function doesn't exist on a given OS, that function is stubbed on that OS anyway—the header file still contains the prototype, the library file still contains the import reference, and the DLL still contains the entry point. For most cases in which the function is unsupported on a given OS, the function fails and sets the last error to ERROR_NOT_IMPLEMENTED.

There are a number of ways to ensure that the functions you are using are available on a given OS:

- All the new Win95 functions that are not supported on Windows NT are inside a **#if WINVER >= 0x0400** check in the header files. If you set **WINVER < 400** when you are compiling, you will not get prototypes defined for the new functions.
- You can use the PORTTOOL to determine which functions in your source code are supported only on a given platform.
- On each function's reference page, the documentation contains a QuickInfo pop-up window, indicating which platforms are supported and any platform-specific differences.
- You can do run-time checks to see whether functions are implemented, and execute only certain code paths on certain operating systems.

Documentation

All the documentation for Win32 programming is now consolidated into one reference. The following features make the differences between the platforms clearer:

- The QuickInfo hot spot on the reference page for each Win32 function is a pop-up window indicating which operating systems support a given function, when the function was first introduced, and whether there are any platform-specific notes for it.
- In each reference page, bold tags highlight platform differences such as flags that are or are not supported, and any subtle implementation inconsistencies. The bold tags indicate the operating system for which the differences apply.
- All the functions that are new for Windows 95 and not yet implemented for Windows NT are marked as "(New - Windows 95)."

Samples

Eighty percent of the samples in the Win32 SDK will run on both Windows 95 and Windows NT 3.51. The remaining 20 percent, which exploit certain functional areas like OpenGL, Security, and Shell Extensions, have been broken down into \WINNT and \WIN95 subdirectories within the samples tree. All these samples get installed no matter which operating system is the host. Thus, you can easily target an application for either or both operating systems, independent of the host environment.

Remote Procedure Call Support

Microsoft RPC is a toolkit for developing distributed applications in C/C++. The toolkit includes:

- MIDL compiler for Windows NT.
- C/C++ language header files and run-time libraries for Windows NT, Microsoft Windows 95, Microsoft Windows 3.*x*, MS-DOS, and Apple Macintosh.
- Sample programs for Windows NT, Windows 95, Windows 3.*x*, MS-DOS, and Apple Macintosh.

RPC Installation

The Win32 SDK provides the components of the RPC toolkit as part of its standard installation. No additional installation is required for 32-bit RPC development.

To produce MS-DOS and Windows 3.*x* RPC clients, install version 1.50 of the Microsoft Visual C/C++™ development environment, and then install the MS-DOS/Win16 RPC toolkit from disk. To install the MS-DOS/Windows 3.*x* version of the RPC toolkit, use the Setup program located in the MSTOOLS\RPC_DOS\DISK1 directory.

To install the Macintosh version of the RPC toolkit, use the Setup program in the MSTOOLS\RPC_MAC\DISK1 directory. To produce Apple Macintosh clients, install Microsoft Visual C/C++ for the Macintosh and run the MIDL compiler on Windows NT.

Because the 16-bit MIDL compiler is no longer supported, we strongly recommend that you develop applications using the 32-bit MIDL compiler. However, to write MS-DOS or Windows 3.*x* - based applications without a 32-bit C/C++ compiler that is capable of targeting MS-DOS, you must compile the IDL file with a 32-bit MIDL compiler on Windows NT or Windows 95. Then compile the application and stubs using your C/C++ compiler. To write MS-DOS or Windows 3.*x* - based applications using a 32-bit C/C++ compiler that is capable of targeting MS-DOS, compile both the IDL file and C/C++ files on Windows NT or Windows 95.

RPC Documentation

Documentation for RPC is available in the RPC section of the Browser and includes conceptual material, MIDL language and command-line references, and run-time API references.

RPC Samples

RPC sample program source files are available in the directory MSTOOLS\SAMPLES\RPC. The README.TXT file in that directory describes each sample.

RPC Redistributables

For information about redistribution, see <*CD_DRIVE*>\DOC\REDIST\REDIST.TXT.

OLE

OLE is a set of integration technologies that enable your application to work with other applications. OLE is built on top of the Component Object Model, a robust and extensible object model that enables OLE components to work together. OLE Documents enables you to add support to in-place activation and linking to your compound document applications. OLE Storage lets you store your applications data hierarchically, and provide summary information and properties that the operating system can utilize. OLE Automation lets you create a programmatic interface to your application, so that it can be driven by another application. OLE Drag and Drop lets you add drag-and-drop support to your application.

OLE Documentation

Documentation for OLE is available in the OLE section of the Browser.

OLE Samples

OLE samples are available in the \MSTOOLS\SAMPLES\OLE directory. For further details on the samples, see the README.TXT in that directory.

Win32s

In \MSTOOLS\WIN32S you will findWin32s version 1.30. New features in this version are:

- Support for the new Common Controls and Common Dialogs
- Support for the OLEDLG functions
- Support for WinHelp 4.0
- Support for the rich edit controls
- Size and speed enhancements

Multimedia

The Win32 SDK now includes full and up-to-date support for multimedia functions. This includes the headers, libraries, samples, and documentation for Video for Windows 1.1-equivalent functionality, including Audio Compression Manager 2.0 and Video Capture. The Win32 SDK continues to include support for functionality equivalent to the Windows 3.1 16-bit API, as also found in Windows NT 3.1, including Wave Audio, MIDI, and MCI device control.

Multimedia Documentation

Documentation for multimedia is available in the Multimedia section of the Browser.

Multimedia Samples

Multimedia samples are available in the \MSTOOLS\SAMPLES\MM directory. For further details on the samples, see the README.TXT in that directory.

InstallSHIELD Setup Toolkit

For this release of the Win32 SDK, Microsoft has licensed the InstallSHIELD SE Setup Toolkit to help you quickly create setup applications that adhere to the Microsoft Windows Application Setup Guidelines.

InstallSHIELD SE provides a quick and easy way for Win32 developers to create simple, functional setup programs. InstallSHIELD SE uses a standard Windows 95 user interface-style setup. InstallSHIELD SE strictly conforms to Microsoft's Windows Application Setup Guidelines for Independent Software Vendors. It contains only features and functions that are appropriate for Win32-based applications. Dialog boxes, progress indicators, and error messages are predefined and built-in.

InstallSHIELD SE includes an integrated uninstallation program that works in conjunction with the Setup Toolkit. The uninstallation program allows your end users to cleanly and effectively uninstall applications and recover system resources such as disk space used by files and registry entries.

To install the InstallSHIELD Setup Toolkit, go to the \MSTOOLS\ISHIELD*platform*\DISK1 directory of the CD-ROM and type **setup.**

32-bit Microsoft Setup Toolkit

The Setup Toolkit allows you to write 32-bit setup programs for your Win32-based applications and Windows NT device drivers in a straightforward manner. Among the toolkit's features are the ability to include your own dialog boxes in the program, and to include your own setup logic through the use of C-language installation scripts. The Microsoft Setup Toolkit now supports Uninstall and Silent Install. Disk layout is also supported through a graphical interface that allows you to specify the list of files and information for each file, such as which disk each file must go on, whether the file should be compressed, and so on. The toolkit will then automatically compress all the necessary files and lay them out for floppy disk, CD-ROM, or network installation.

This Setup Toolkit is provided as a Microsoft-supplied alternative to the InstallSHIELD setup toolkit.

Information on how to install 16- and 32-bit applications that take advantage of user profiles and the Registry (allowing remote upgrading) are discussed in another setup document in \MSTOOLS\MSSETUP\SETUP.WRI.

Setup Toolkit Documentation

Documentation for the Setup Toolkit is available in the Browser in the Setup Toolkit section.

OpenGL (Windows NT)

Windows NT Workstation version 3.51 includes high-performance 3-D graphics capabilities as a native part of the operating system. These capabilities are provided with the OpenGL API. OpenGL is an operating-system- independent, industry-standard library of graphics functions originally developed by Silicon Graphics, Inc., (SGI) on their workstation products. Microsoft has licensed this technology from SGI to provide these powerful 32-bit functions in Windows NT. OpenGL is defined by an Architecture Review Board consisting of Digital Equipment Corporation, IBM, Intel, Microsoft, and Silicon Graphics.

These advanced graphics capabilities are extremely important for users of CAD/CAM; multimedia authoring tools; industrial, interior, and mechanical design; statistical analysis; and scientific products in which visualization of designs and data is important. It is also used for 3-D modeling and animation applications. This technology was used in recent commercial films such as *Jurassic Park* and *Terminator 2*.

OpenGL Documentation

Documentation for OpenGL is available in the OpenGL section of the Browser; it includes conceptual material, a porting guide, and reference material.

OpenGL Samples

Examples of the new, advanced three-dimensional graphics capabilities are located in the \MSTOOLS\SAMPLES\OPENGL directory. For more information, see the README.TXT file in that directory.

POSIX Development (Windows NT)

All the necessary headers and libraries for building POSIX applications have been included in this release of the Win32 SDK. These files are located in the \MSTOOLS\POSIX directory. To set up your environment variables for POSIX, type **setnvpsx <***sdkroot***>** where *sdkroot* is the path to the POSIX directory. For example, this could be **setnvpsx c:\mstools**.

POSIX Samples

In the MSTOOLS\POSIX\SAMPLES\PSXARC directory is a sample POSIX application. This sample uses definitions for POSIX flags and libraries located in the \MSTOOLS\INCLUDE\WIN32.MAK file.

There are no debuggers provided for POSIX development.

Software Compatibility Test Kit (Windows NT)

The Win32 SDK includes a Software Compatibility Test to help ensure that your existing MS-DOS and Windows 3.1-based applications run well on Windows NT. For more information, see the *Software Compatibility Test* help file (SCT.HLP).

Pen Services API (New - Windows 95)

A subset of Windows 95 Pen Services is available on every computer that runs Windows 95. Functions in this library allow any Win32-based application to display and manipulate pen data (also called "ink") that was originally collected on a pen-enabled system. A few practical applications for these services might include:

- Using Windows applications to display a signature
- Verification of signatures collected on a pen-based mobile computer
- Animation of ink for presentations or games
- Merging, compressing, and storage of ink collected on pen-based systems
- Displaying graphics, maps, or handwritten notes that have been drawn on a pen-based system

Please note that at this time the Pen API is available only on Windows 95. It will be available on Windows NT in a future release.

Pen Services Documentation

Documentation for pen services is available in the "Pen" and "Guide to Programming Windows 95" sections of the Browser; these sections include conceptual and reference material.

Pen Services Samples

Examples of the new pen capabilities are located in the \MSTOOLS\SAMPLES\WIN32\WIN95\ {HFORM|INKPUT|PKPD} directories.

Telephony API (TAPI) (New - Windows 95)

The Telephony API (TAPI) provides a standard API for the setup and management of telephone networks. TAPI allows applications developers to programmatically control telephone networks, such as dialing, transferring or terminating calls, and also provides access to information from the network, such as Caller ID. TAPI lets the operating system manage telephone-line resources and arbitrate usage amongst a variety of voice, fax, data, or even video applications. TAPI also virtualizes the telephone network, allowing applications to run against the myriad of different types of phone networks (for example, analog, PBXs, ISDN, CENTREX, cellular) without modification to the application. Simply add the appropriate driver. TAPI provides the "glue" to integrate the two most common business tools: the PC and the telephone.

Please note that at this time the Telephony API is only available on Windows 95. It will be available on Windows NT in a future release.

TAPI Documentation

Documentation for TAPI is available in the Telephony Application Programming Interface section of the Browser; it includes conceptual and reference material.

TAPI Samples

An example of TAPI is located in the \MSTOOLS\SAMPLES\WIN32\WIN95\TAPICOMM directory. For more information, see the README.TXT file in that directory.

Messaging API (MAPI) (New - Windows 95)

The Microsoft Messaging Application Programming Interface (MAPI) provides a modular architecture for building powerful messaging-based applications and service providers. The architecture specifies several well-defined components to allow administrators to mix and match components to support a broad range of vendors, computing devices, and communication protocols. This makes it possible to use the MAPI architecture for e-mail, scheduling, personal information managers, bulletin boards, and online services.

Please note that at this time MAPI is available in retail form on Windows 95 only. The MAPI runtimes, headers, and libraries for Windows NT are still in beta form, and cannot be redistributed.

MAPI Documentation

Documentation for MAPI is available in the MAPI section of the Browser; it includes conceptual and reference material.

MAPI Samples

Examples of MAPI are located in the \MSTOOLS\SAMPLES\MAPI directory. For more information, see the README.TXT file in that directory.

Shell Extensions (New - Windows 95)

In Windows 95, applications can extend the shell in a number of ways. A shell extension enhances the shell by:

- Providing additional means of manipulating file objects.
- Simplifying the task of browsing through the file system and networks.
- Giving the user easier access to tools that manipulate objects in the file system.

For example, a shell extension can assign an icon to each file or add commands to the context menu and file menu for a file. Windows 95 supports the following types of shell extensions:

- Context menu handlers
- Drag-and-drop handlers
- Icon handlers
- Property sheet handlers
- Copy hook handlers

Please note that for this release, Windows NT does not officially support the new Windows 95 User Interface or Shell. However, Microsoft has provided a "Shell Technology Preview" to all MSDN Level II subscribers. This Preview is intended to enable software developers and vendors to test their applications on both Windows 95 and Windows NT version 3.51 platforms. This user interface will be integrated into the base Windows NT product and made available in a future release of Windows NT. The date of this release is to be determined.

For information regarding the Shell Technology Preview, please see README.WRI in the *platform*\NEWSHELL directory of the Windows NT 3.51 Operating System CD you received.

Shell Extension Documentation

Documentation for shell extensions is available in the Browser in the Win32 reference, as well as in the section "Guide to Programming Windows 95"; the documentation includes both conceptual and reference material.

Shell Extension Samples

Examples of shell extensions are located in the \MSTOOLS\SAMPLES\WIN32\WIN95 directory.

Building Win32-Based Applications

The Win32 SDK provides a 32-bit development environment. Most of the tools are very similar to their 16-bit equivalents, which are included in the Windows 3.1 SDK. These tools represent the basic level of functionality required to create a Win32-based application. Many of the components are character-mode applications, which are used individually or with makefiles. The Win32 SDK provides a portable and consistent development environment for the *x*86, MIPS, Alpha AXP, and PowerPC platforms. It does not provide a graphical integrated development environment. If a tool is supported only on a particular operating system, it is marked as such.

This section highlights the tools provided in the SDK. It is divided into the following sections:

- Writing your application
- Compiling your application
- Debugging your application
- Tuning your application
- Samples

Writing Your Application

The first step in building a Win32-based application is to write the source code and create the resources required for your program. Sample code, resource manipulation tools, WinDiff (a file comparison utility), and a porting tool for converting your 16-bit applications are all provided.

Editor (MEP.EXE)

The 32-bit Microsoft Editor is a user-configurable character-oriented text editor. You can customize the editor by editing the TOOLS.INI file in the \MSTOOLS\INIT directory and ensuring that your INIT environment variable points to this directory. Key bindings are available to emulate BRIEF, Epsilon, and the Microsoft QuickBasic®/QuickC® editors.

Online Help is available from within the editor.

To gain access to the editor's online Help

1. Set the INIT environment variable to point to the location of TOOLS.INI.
2. Make sure that MEP.HLP is located in your path.

   The Win32 SDK setup program installs MEP.HLP in \MSTOOLS\BIN by default.

To enable the mouse so that it can be used to select text and position the cursor, add the following lines to TOOLS.INI:

```
[m mep]
usemouse:yes
```

To enable other popular alternate key bindings, replace the existing TOOLS.INI file with either BRIEF.INI, EPSILON.INI, or QUICK.INI.

Dialog Editor (DLGEDIT.EXE)

The Dialog Editor is used to construct dialog boxes for your application. The Dialog Editor contains its own online reference information available for help.

The Dialog Editor can also handle user-defined custom controls. For more information on creating custom controls, see CUSTCNTL.TXT in the <CD DRIVE>:\DOC\FILEFRMT directory, along with the **spincube** sample application.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Icon, Bitmap, and Cursor Editor (IMAGEDIT.EXE)

The Image Editor creates bitmaps, icons, and cursors that you can use as resources in your windows applications. Complete help is available online.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Font Editor (FONTEDIT.EXE)

The Font Editor lets you create your own custom bitmap fonts. Complete help is available online. Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Porting Tool (PORTTOOL.EXE)

The Porting tool helps you automate the task of converting 16-bit Windows application source code to Win32. Although all efforts have been made to maintain a similar API set between 16-bit and 32-bit Windows, there are some modifications that must be made to existing code. The Porting tool processes all your source files, either interactively or in background mode, and highlights those areas that need to be addressed. It also provides guidelines as to what changes may be necessary. The Porting tool has also been enhanced to highlight functions that are different between Windows 95 and Windows NT, as well as Win32s.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Hotspot Editor (SHED.EXE)

Windows NT Only

The Hotspot Editor creates segmented hypergraphics. A segmented hypergraphic is a graphic (bitmap or metafile) that has embedded hyperlinks that users choose to initiate an action such as a jump to another help topic. Use the Hotspot Editor to create graphics to add to the help files for Windows Help. This utility is used in conjunction with the help compilers described in Help Compilers (HC31.EXE/HCW.EXE) later in this section. Complete help is available online.

File Comparison Utility (WINDIFF.EXE)

The File Comparison utility is a powerful graphical comparison program capable of comparing files as well as directory trees. It highlights both missing and differing files, and provides a graphical expansion of files that differ for quick comparison. You can also launch the editor of your choice from WINDIFF to edit the composite of the two files. This tool is useful for source-code maintenance and file comparison in large software projects.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Screen Viewer (ZOOMIN.EXE)

The Screen Viewer utility enables you to zoom in on any portion of the screen, providing a magnified view of whatever is under the selection box. You can use it to verify accurate alignment of bitmaps and text, and to note inconsistencies in icons, cursors, dialog boxes, and so on. You can copy the enlarged area to the clipboard and paste it to any Windows drawing package.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Video Capture (VIDCAP32.EXE)

With the Video Capture tool, you can digitize source video from supported video capture cards. Either ask your capture-board vendor for Windows NT 3.51 drivers, or look on the Windows Driver library for driver support. You may find VIDCAP32 more useful when you use it in conjunction with some of the compiled samples, particularly "AVIEdit."

Compiling Your Application

Once the code is written and the resources are ready, it is time to build the application.

NOTE The compiler, linker, and librarian documented in the following two sections are provided on the PowerPC platform only. The remaining sections apply to all four platforms. To build your applications on *x*86, MIPS, and Alpha AXP platforms, you must have Microsoft Visual C++ 2.*x* for Windows NT, or a compatible 32-bit compiler product.

C/C++ Compiler (CL.EXE)

Windows NT Only

A 32-bit equivalent of the Microsoft C/C++ compiler is provided for the PowerPC platform. This compiler is ANSI compliant, and is front-end-compatible with the Microsoft VC++ X86, MIPS, and Alpha compilers, allowing you to maintain a single code base for multi-platform development.

Linker and Librarian (LINK.EXE/LIB.EXE)

Windows NT Only

The Linker is based on the UNIX COFF standard with Microsoft extensions for PowerPC only.

Standardized Makefiles (WIN32.MAK)

To maintain a single code base and ease porting between the various architectures, a standardized make include file is provided to simplify creating and maintaining your own project makefiles. You can use this makefile to create a single application that will run on both Windows 95 and Windows NT.

WIN32.MAK provides environment variables such as *cflags*, *cdebug*, and *ldebug*, which act as default switches for the compiler and linker. Variables such as *guilibs* and *conlibs* provide the necessary library lists for creating most standard GUI and Console-mode applications. By using WIN32.MAK as the foundation for your own makefiles, global changes for optimization or new platforms become simple. Review the sample makefiles provided with the Win32 SDK that illustrate NMAKE and WIN32.MAK macro usage.

Resource Compiler (RC.EXE)

The Resource Compiler included with the Win32 SDK is source-compatible with the Microsoft Windows 3.1 Resource Compiler. It reads the same resource script (.RC) files but produces 32-bit resource files that can then be read and edited by the Dialog Editor.

For a complete description of the resource file format, see RESFMT.TXT in the <*CD_DRIVE*>:\DOC\FILEFRMT directory.

Message Compiler (MC.EXE)

The Message Compiler creates message files that can then be used in event logging.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Help Compilers (HC31.EXE/HCW.EXE)

HC31.EXE builds help files that can be used with Microsoft Windows NT and Microsoft Windows 3.1 WinHelp. The Help Compiler supports metafiles, tables, segmented hypergraphics, secondary help windows, macros, and many other features documented online in the Tools User Guide section of the Browser.

HCW.EXE creates help files that can be used with Microsoft Windows WinHelp 4.0. This tool is still in its preliminary form, and will be updated when Windows 95 is released.

Multiple-Resolution Bitmap Compiler (MRBC.EXE)

The Multiple Resolution Bitmap Compiler creates bitmap files that contain several bitmaps representing the same image, but having different resolutions, aspect ratios, or color formats. Use the compiler to create bitmap resources to add to help files for Windows Help. When a help file contains a multiple-resolution bitmap, Windows Help displays only the bitmap that best matches the characteristics of the current display device.

Make Typelib (MKTYPLIB.EXE)

The Make Typelib generates type libraries from ODL files. Type libraries are used for OLE Automation.

Microsoft Interface Definition Language Compiler (MIDL.EXE)

MIDL is the IDL compiler. It processes IDL files and generates RPC stubs. It is used for both RPC and defining your own custom interfaces for OLE. For further information, see the RPC section of the Browser.

Windows 95 Thunk Compiler (THUNK.EXE)

Windows 95 Only

The Microsoft Windows Thunk Compiler lets you create thunks between 16- and 32-bit code. For more information regarding thunks on Windows 95, see the Guide to Programming Windows 95 in the Browser.

Debugging Your Application

Once you have written and built your application, the next step is to debug it. The Win32 SDK provides several debuggers and debugging tools to help isolate problems.

Windows Debugger (WINDBG.EXE)

The Microsoft Windows Debugger is a powerful C/C++ graphical debugger. It makes use of MDI to provide immediate access to windows displaying dynamically updated information about single- or multithreaded programs. The Debugger includes:

- C/C++ support
- Remote debugging to a second Windows NT computer, Windows 95 computer, or a Windows 3.1 system running Win32s
- Cross-platform debugging
- Assembly language debugging
- 16-bit debugging; multiprocess support
- Kernel debugging
- Online Help

The Debugger also features windows for:

- Source- and assembly-level debugging
- Display of local variables
- Watchpoints
- Breakpoints
- Memory dumps
- Machine registers
- Call stacks

Spy (SPY.EXE)

The Spy utility allows the developer to monitor messages being passed in the system and between applications. The Spy utility can log all messages between processes, as well as filter specific messages.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

DDESpy (DDESPY.EXE)

The Dynamic Data Exchange Spy utility lets you view transactions that are occurring between DDE-based client-server processes running in the system.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Process Viewers (PVIEW.EXE/PVIEW95.EXE)

The Process Viewers are tools for viewing many aspects of the processes running in the system. All information is provided on a per-process basis, including total memory commitment, working set, pool usage, and CPU time. Each process is enumerated on a per-thread basis, illustrating individual thread priorities, memory usage, and processor times.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Process Walker (PWALK.EXE)

Windows NT Only

The Process Walker provides a complete listing of the memory usage of a particular process. It displays the process's entire user-mode address space, indicating free, committed, and reserved memory, as well as which DLLs are loaded and where the code, data, and stack are located. This tool is extremely useful for tracking down memory leaks and memory usage.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

Heap Walker (HEAPWALK.EXE)

Windows 95 Only

The Heap Walker (HEAPWALK.EXE) lets you examine the global heap (the system memory that the Microsoft Windows operating system uses) and local heaps used by active applications and dynamic-link libraries (DLLs) in your Windows system. Heap Walker is useful for analyzing the effects your application has when it allocates memory from the global heap or when it creates user interface objects or graphics objects.

Dispatch Tester (DISPTEST.EXE)

The Dispatch Tester is a test engine for applications that support OLE Automation.

Data Object Viewer (DOBJVIEW.EXE)

The Data Object Viewer is a test and diagnostic tool for viewing objects that support the **IDataObject** interface.

Doc File Viewer (DFVIEW.EXE)

The Doc File Viewer displays a tree representation similar to the directory view of File Manager. Streams can be opened and the data they contain can be browsed.

Source code is provided in the \MSTOOLS\SAMPLES\OLE directory.

Running Object Table Viewer (IROTVIEW.EXE)

The Running Object Table Viewer gives a snapshot display of all objects registered with OLE's running object table.

OLE Viewer (OLE2VW32.EXE)

The OLE Viewer is a general-purpose OLE viewer. It is a powerful development and diagnostic tool that gives access to many different types of information that would be of interest to someone developing an OLE application.

Source code for DEFO2V.DLL, a user interface extension viewer, is provided in the \MSTOOLS\SAMPLES\OLE directory.

OLE Test Applications (CL32TEST/SR32TEST.EXE)

The OLE test applications are generic 32-bit OLE containers and servers. These are designed to help you test your 32-bit OLE implementation.

Address Book Viewer (ABVIEW32.EXE)

The Address Book Viewer is a tool used to view and edit the MAPI address book entries.

Message Store Viewer (MDBVU32.EXE)

The Message Store Viewer is a tool used to traverse, view, and edit the structure of a MAPI store. This includes stores, folders, messages, and their associated properties.

Forms Installer (PDKIN32.EXE)

The Forms Installer is a tool used to programmatically install MAPI forms into the "Application" forms registry.

MAPI Test Application (SEND32.EXE)

The MAPI Test Application is used to send messages of various types and sizes. This is useful for MAPI developers when testing the functionality of their MAPI providers.

Service Manager (SC.EXE)

Windows NT Only

The Service Manager is a tool that drives services. With the Service Manager you can easily start, stop, pause, and dynamically configure your services.

Symbol Editor (SYMEDIT.EXE)

The Symbol Editor is a tool that provides a number of executable manipulation routines. It adds CodeView® information to a Windows NT executable image. It strips all debugging information from an executable and splits the debugging information into a separate .DBG file. See Debug Symbols for information on .DBG files.

Source code is provided in the \MSTOOLS\SAMPLES\SDKTOOLS directory.

MAPSYM (MAPSYM.EXE)

Windows 95 Only

MAPSYM converts the contents of your application's symbol map (.MAP) file into a form suitable for loading with the 80386 Debugger. MAPSYM then copies the result to a symbol (.SYM) file.

80386 Debugger (WDEB386.EXE)

Windows 95 Only

The Microsoft Windows 80386 Debugger (WDEB386.EXE) is used to test and debug Windows applications and dynamic-link libraries (DLLs) running with the Microsoft Windows 95 operating system. With 80386 Debugger commands, you can inspect and manipulate test code and environment status, install breakpoints, and perform other debugging operations.

Debug Symbols (.DBG files)

Windows NT Only

Included with every retail Windows NT operating system CD-ROM are the debugging symbols for every 32-bit binary in the system, including executables, DLLs, and drivers.

The SYMEDIT tool described earlier provides an easy way for application writers to take advantage of this debugging support. SYMEDIT removes all the debugging information from an executable and places it in a .DBG file. This file should remain either in the same directory as the binary, or in the appropriate subdirectory off %SYSTEMROOT%\SYMBOLS. You can then run optimized applications but still have symbolic information available for debugging in the event of a failure.

To aid debugging Win32-based applications, the Win32 SDK setup program automatically installs the .DBG files for the most commonly required user-mode system DLLs.

Debug DLLs (.DLL files)

Windows 95 Only

Included with the Win32 SDK are the debug versions of the most commonly used user-mode components of the Windows 95 operating system, including executables, DLLs, and drivers. These debug binaries allow you to test applications under a "Debug" version of Windows 95. This debug version provides information, or RIPs, indicating when the application has done something illegal, like forgetting to free a resource or passing an invalid parameter.

To switch between retail and debug versions of Windows 95, simply click on the appropriate icon in the Win32 SDK Tools folder.

Kernel Debuggers (I386KD.EXE/MIPSKD.EXE/ALPHAKD.EXE/PPCKD.EXE)

For device-driver debugging, the kernel debuggers have also been provided. You can run these debuggers remotely on either an *x*86-, MIPS-, Alpha-, or PowerPC-based Windows NT system. For more information, see either the Tools User Guide section of the Browser or, for information on connecting and configuring the kernel debuggers, Setting Up KD.

Command Line Debuggers (NTSD.EXE/CDB.EXE)

Windows NT Only

For simple command-line debugging, you can use NTSD on your application. If you don't want to create an additional window for the debugger, use CBD attached directly to the process.

Debug Monitor (DBMON.EXE)

Windows NT Only

The Debug Monitor runs in its own console window and traps messages to OutputDebugString from your application. This means that you no longer have to run a separate kernel debugger just to get this information.

Object Monitor (WINOBJ.EXE)

Windows NT Only

The Object Monitor is provided as a graphical way to watch all the objects in the system and determine relevant information such as description, attributes, and resource usage.

Tuning Your Application

After your application is built and debugged, the final step is to tune your application for optimum performance.

Working Set Tuner (WST)

Windows NT Only

The *working set* of a program is a collection of those pages in the program's virtual address space that have recently been referenced. As the working set size decreases, memory demand decreases. Because it is advantageous to minimize memory demand, the Working Set Tuner measures your application's working set and produces an ordering of the functions (procedures) within the code. Relinking your application with this ordering helps to minimize its working set.

The Working Set Tuner repacks code based on the frequency at which functions are executed, not on what source or object file the code resides in.

For further information regarding Working Set Tuning, see the WST.TXT file in the MSTOOLS\BIN directory. WIN32.MAK also contains macros to help build applications for Working Set Tuning. Typing **nmake tune=1** automatically sets the proper compiler and linker switches and link with WST.LIB.

Call Attributed Profiling (CAP)

Call Attributed Profiling allows you to monitor every function call made by your application, providing both frequency and timing information for each routine. This program helps identify bottlenecks in your application, so that these routines can be optimized or redesigned where necessary.

For further information regarding Call Attributed Profiling, see the CAP.TXT file in the MSTOOLS\BIN directory. WIN32.MAK also contains macros to help build applications for call profiling. Typing **nmake profile=1** will automatically set the proper compiler and linker options.

Windows Function Profiling (WAP)

Windows NT Only

A third method of profiling your applications can be achieved tracing only the Windows function calls. Through WAP you can log all Win32 function calls and determine whether a particular function is either taking an extended amount of time or being called too frequently. For complete details on Windows function profiling, see the WAP.TXT file in the \MSTOOLS\BIN directory.

API Logger (LOGGER32)

Windows NT Only

Logger is a tool that records the calls an application makes to the Win32 API as well as callbacks that the Win32 system makes to the application. The output file produced by Logger is a list of the functions called, the parameters passed to each function, and the return value from each function. Logger is also capable of timing these events and placing this information into the output file. For further information, see the README.TXT file in the \MSTOOLS\BIN\LOGGER32 directory.

Binding (BIND.EXE)

When development of your application is complete, you can bind your executable to get the best performance out of calls to your own DLLs or the system (Win32 API). In general, DLLs include two sets of import address tables: one table for entrypoint names for application use, and another containing the actual address of the entrypoints. Calls to DLL exported functions are made by doing an indirect jump through this table. The Bind utility actually precomputes these addresses, allowing the loader to bypass entrypoint lookup. By binding your executable with all your DLLs and the system DLLs, you can significantly improve the load time of your application.

Rebasing (REBASE.EXE)

The base address of a DLL is specified at link time and is where the loader attempts to place the DLL in memory. If a DLL cannot be loaded at its base address (memory already occupied), the loader places the DLL elsewhere in virtual memory. Fixups must then be performed on all calls into the DLL, slowing down the load time. In the case of large applications with many DLLs, it is important that each DLL have a different base address to minimize load time. REBASE has been provided to change DLL base addresses.

The system DLLs are currently based from 0x70000000 - 0x78000000 (0x68000000 - 0x78000000 on MIPS). Application writers are encouraged to base their DLLs from memory location 0x60000000 to 0x68000000, which equals 128 MB of address space. To help avoid collisions, you can base your DLL on the first character of the name A - C at 0x60000000, D - F at 0x61000000, and so on.

Performance Monitoring Tools

Windows NT Only

The Win32 SDK also provides a variety of tools for monitoring system, process, and memory usage. Tools such as PSTAT.EXE, PERFMTR.EXE, WPERF.EXE, TOP.EXE, VADUMP.EXE, and MEMSNAP.EXE perform various monitoring functions.

Samples

The Win32 SDK provides a number of sample applications located in the \MSTOOLS\SAMPLES directory. Review the file named Samples Help, which discusses important samples, highlighting their content and usage.

You can install any combination of the following categories of samples, except Q_A, which remain at the root of the CD-ROM:

Category	Description
Win32	Examples of Win32 functionality, such as threads, flat memory model, and so on
Frmwork	New source code, compatible with Windows 95, that demonstrates new common controls
OLE	Examples of the new 32-bit OLE, including automation
OpenGL	Examples of the new advanced 3-D graphics capabilities
SdkTools	Source code to many of the tools provided in the Win32 SDK
RPC	Examples of remote procedure calls and custom interfaces
MAPI	New Messaging API samples
MM	New multimedia samples
Q_A	Sample code for frequently asked questions

Getting More Information

Along with the development tools, a vast amount of information is available to help you develop Win32-based applications. More than 30 MB of online help and two forums on CompuServe are dedicated to the Win32 SDK. The WINOBJ forum is specific to OLE. The MSWIN32 forum is for Win32. MSDN phone support is available at 1-800-936-5800.

Online Help Files

The Win32 SDK contains more than 7,500 pages of online documentation in Browser format, covering everything from the Win32 API to the message compiler.

The following major topics are contained in the Browser help file, and are fully cross-indexed and searchable.

- Win32 API Reference
- Video For Windows Reference
- OpenGL
- RPC Programming Guide and Reference
- Win32s Programmer's Reference
- 32-bit Setup Toolkit Reference
- Programming Techniques
- Guide to Programming Windows 95 (New - Windows 95)
- MAPI Reference (New - Windows 95)
- TAPI Reference (New - Windows 95)
- PEN Reference (New - Windows 95)
- Tools User Guide
- Microsoft Knowledge Base for the Win32 SDK

The following ship as standalone Help files as well:

- 80386 Debugger Reference (WDEB386.HLP)
- Samples Quick Reference (SAMPLES.HLP)
- Tools Quick Reference (PowerPC only—TOOLSQ.HLP)
- C/C++ Reference (PowerPC only—MSC.HLP)
- OLE Tools Reference (OLETOOLS.HLP)
- Internet Short Cut Reference
- UI Style Guide

Supplemental Information

Further information on the following topics can be found in the <*CD_DRIVE*>\DOC directory. For details, see the README.TXT file.

- Custom Controls
- The Resource File Format
- Enhanced MetaFile Format
- Hardware Compatibility List for Windows NT 3.51
- Cross-platform Win32 Compatibility
- 16-bit Thunking Interface
- Sockets Multicasting Support
- Posix Conformance Document
- SNMP Programmer's Reference
- AppleTalk® Programmer's Reference
- 16-bit Debugging Support
- Windows 95 SEH

Using the MSDN Browser

The Browser opens automatically when you double-click the Win32 SDK online reference icon. When you double-click a book icon in the Browser, it opens to reveal the next level of information—either topics or more books. To open a topic, double-click its icon in the Browser.

Use the Keyword Index button to find topics listed by title, and use the Query button to perform full-text search. This is often the fastest way to find reference topics for functions and related information. The entire set of books has been cross-indexed to make navigation even easier.

The Group and Overview buttons provide further navigational assistance. They are active for selected reference topics in the Win32 SDK documentation. Click the Group hot spot to find listings of functions, messages, or structures related by functionality to the topic displayed on the screen. Click the Overview hot spot to see a discussion of the functionality relating to the displayed reference topic.

The Browser also has its own online Help, available from the Help menu.

How to Use the Windows NT™ Trademark of Microsoft Corporation

1. OUR MARKS ARE IMPORTANT

Microsoft Corporation's trademarks and service marks are valuable corporate assets. This guide will assist you in the proper use of the Windows NT™ trademark.

2. PRODUCT NAMES

Without a specific trademark license from Microsoft, Microsoft's marks **MAY NEVER** be incorporated as part of the name of a product or service of another company. You may not include "Windows," "Windows NT," "NT" or any potentially confusing variation in the name of your product. This same rule applies to other Microsoft marks, such as **Microsoft®**, which should never be included in your product name or company name.

3. PACKAGING AND ADVERTISING

The compatibility of an application program with the Windows NT operating system may be noted on packaging, collateral material or advertising (***but not included in the product name***) by using phrases such as "for use with," "compatible with" or "for" the Windows NT operating system. On all such materials your name must appear more prominently than the Windows NT mark **and** the Windows NT mark should be visually distinguished from your product name by putting it in a different font, or color, or on a different line. This is important to avoid any implication that your product is produced, endorsed or supported by Microsoft. Here are some examples:

Do Say:	***XYZ*** for the Windows NT™ operating system
Do Not Say:	*XYZ* Windows NT
	Windows NT *XYZ*
	NT *XYZ*
	XYZ NT

Correct reference to the Windows NT mark includes:

(i) using the symbol "™" with our mark Windows NT and use "®" with Microsoft. The symbols should be placed next to the last letter of the mark, at the upper right or at the baseline. In advertising copy, the proper symbol should be used at the first or most prominent mention;

(ii) placing the generic product name, "the descriptor," immediately after the mark. The descriptor for the Windows NT operating system is "operating system"; and

(iii) including the following notice on such material:

"Windows NT is a trademark of Microsoft Corporation."

4. ALWAYS USE OUR MARKS AS PROPER ADJECTIVES

Trademarks identify a company's goods or services. A trademark is a proper adjective that modifies the generic name of a product or service. Microsoft is our house mark to identify all products and services which originate from us. Our Windows NT mark identifies our operating system.

Examples:

Proper Adjective	**Generic Name or Descriptor**
Microsoft®	software
Windows NT™	operating system

5. DO NOT COMBINE THE TRADEMARK WITH AN IMPROPER GENERIC NAME

Windows NT is the mark we use to identify our operating system. Application programs designed to run

on the Windows NT operating system are not "Windows NT applications," but rather application programs for the Windows NT operating system.

Examples:

Do Say:	Application(s) for the Windows NT™ operating system
	Windows NT™ - based applications
Do Not Say:	Windows NT™ application(s)
	NT application(s)
Do Say:	user of the Windows NT™ operating system
Do Not Say:	Windows NT™ user

6. DO NOT USE OUR TRADEMARKS AS POSSESSIVES OR IN PLURAL FORMS

Examples:

Do Say:	Scalable fonts in Windows NT™
Do Not Say:	Windows NT's scalable fonts
Do Say:	The company ordered two copies of Windows NT™
Do Not Say:	The company ordered two Windows NTs

7. DO NOT ABBREVIATE OR CREATE ACRONYMS

Our marks should not be abbreviated.

Example:

Do Not Say:	Win NT
	WNT (for Windows NT)

8. COMPANY NAMES

Microsoft marks **MAY NEVER** be incorporated in your company name (***whether a corporate name or d/b/a***).

Do Not Say:	Windows NT, Inc.
	XYZ Windows NT Company

9. THE FLAG LOGO

The Flag Logo of Microsoft is a valuable symbol for our Windows NT™ operating system and related products. The Flag Logo **MAY NEVER BE USED WITHOUT A LICENSE FROM MICROSOFT.** OEMs, ISVs and IHVs are invited to contact Systems Marketing, Windows Logo Department, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, facsimile number (206-936-7329) for details regarding licensing criteria for the Flag Logo for use with products which are "Ready-to-Run" or are compatible with Windows NT™.

Appendix B

How to Use the Windows® 95 Trademark of Microsoft Corporation

Microsoft Corporation's trade marks and service marks are valuable corporate assets. These guidelines will assist you in the proper use of the Windows® trademark and other Microsoft marks.

RULES FOR PROPER USE

1. PROPER USE OF THE WINDOWS® 95 PRODUCT NAME.

The registered trademark symbol "®" should immediately follow the word "Windows." Please be sure to include a space between the trademark symbol and the designation "95." Mark the trademark with the proper notice symbol at the first or most prominent mention.

Correct	Incorrect
Windows® 95	**Windows®95 or Windows®'95**

2. SET TRADEMARKS APART FROM OTHER WORDS OR THE NOUNS THEY MODIFY.

The common way to do this is to properly capitalize the product name and designate the trademarks with the appropriate symbols– ® or (TM). You can also use underlining, italics, or bold type. Examples:

Correct	Incorrect
When you start up the Windows® operating system, click on the...	When you start up Windows, click on the...

3. USE MICROSOFT TRADEMARKS AS PROPER ADJECTIVES.

Trademarks are adjectives that describe a specific brand of product. Because a trademark is an adjective, use it with the appropriate noun that it modifies.

In the case of our Windows trademark, follow the correct example below.

Correct	Incorrect
application for the Windows® operating system	**Windows applications [or machine/PC]**

You may refer to products that are compatible with the Windows operating system as: "Windows®-compatible products" , "Windows®-based products" or "products for Windows®" or "Windows® 95 compatible."

4. NEVER COMBINE A MICROSOFT TRADEMARK/PRODUCT NAME WITH YOUR (OR ANOTHER THIRD PARTY'S) TRADEMARK/PRODUCT NAME.

Trademarks serve to identify the source of a product. If Microsoft trademarks are combined with the trademarks or product names of others, consumers may be confused as to which company is the source of a product.

Correct	Incorrect
XYZ is compatible with the Windows® 95 operating system	**XYZ/Windows® 95 or XYZ Windows**

5. DON'T USE MICROSOFT TRADEMARKS IN THE POSSESSIVE OR PLURAL FORM.

Correct	Incorrect
the Windows interface	**Windows' interface**

6. USE THE APPROPRIATE TRADEMARK SYMBOL IN THE PROPER PLACE AND GIVE PROPER ATTRIBUTION.

Symbols:

® **= registered trademark or service mark**

TM = trademark ownership claimed

Notice:

"_____ is a registered trademark and _____ is a trademark of Microsoft Corporation."

7. DO NOT SHORTEN, ABBREVIATE OR CREATE ACRONYMS OUT OF MICROSOFT TRADEMARKS.

Correct	**Incorrect**
Windows 95 or Windows	**Win95, Win software**

8. MICROSOFT MARKS MAY NEVER BE INCORPORATED IN YOUR COMPANY NAME (WHETHER A CORPORATE NAME OR D/B/A).

USE OF MICROSOFT TRADEMARKS IN ADVERTISING AND ON PACKAGING

- The compatibility of an applications program with the Windows® 95 operating system may be noted on packaging, collateral material or advertising (***but not included in the product name***) by using phrases such as "for use with," "compatible with" or "for" the Windows® 95 operating system. On all such materials your product name must appear more prominently than the Windows mark **and** the Windows mark should be visually distinguished from your product name by putting it in a different font, or color, or on a different line. This is important to avoid any implication that your product is produced, endorsed or supported by Microsoft.
- You must have a distinct product name for your product which does not include any Microsoft product name or trademark.
- You should not market any product under a name that is confusingly similar to a Microsoft product name or trademark.
- You should not place your company name, trademarks, service marks or product names next to a Microsoft product name on packaging, disk labels, or advertisements.
- You must not imitate Microsoft packaging or product logos.

USE OF THE DESIGNED FOR MICROSOFT® WINDOWS® 95 LOGO

The Designed for Microsoft Windows 95 Logo is a valuable trademark of Microsoft and **MAY NEVER BE USED WITHOUT A LICENSE FROM MICROSOFT.** OEM's, ISV's and IHV's are invited to contact Systems Marketing, Windows Logo Department, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, facsimile number (206-936-7329) for details regarding licensing the Designed for Microsoft Windows 95 Logo. This logo may only be used in conjunction with those products that meet the applicable criteria for licensing the Logo.