The Windows 32-Bit API: An Overview

Since its original release in 1985, Microsoft® Windows™ has become the leading graphical system for IBM® compatible personal computers. Version 3.0, released in May 1990, was a milestone that broke the 640K barrier of the Microsoft MS-DOS® operating system by running applications in protected mode, thus making it possible to develop much more sophisticated applications. This innovation spawned myriad applications and is responsible for the huge success of the Windows environment in the marketplace showcased by the volume of graphical applications sold (see Figures 1 and 2).

Between May 1990 and May 1991, more than 4 million copies of Version 3.0 were sold. International Data Corporation estimates that an additional 7.8 million copies will be sold during 1992. In addition, more than 66 thousand Microsoft Windows Software Development Kits version 3.0 have been shipped, a clear indication of the number of applications likely to appear during the next 12 to 18 months. By spring 1991, more than 1200 Windows-based applications were shipping.

Building on this achievement and on the success of independent software developers, Microsoft is extending and expanding the Windows environment so that Windows-based applications can run on a broad range of computing platforms — from low-end battery-operated portables to high-end RISC workstations and multiprocessor servers.

We are expanding Windows to make it fully 32-bit, and are adding additional operating system services. Microsoft Windows for Pen Computing and Microsoft Windows with Multimedia Extensions will also take advantage of new hardware technologies.

Windows Today

Today many people think of Windows as a graphical add-on to the familiar MS-DOS operating system they have used for years. This perception took much of the fear out of upgrading to Windows for the end user. But in fact, Windows is not limited by MS-DOS.

Windows is a complete operating system that provides extra features on top of MS-DOS and replaces certain MS-DOS features. Windows Version 3.0 does not use MS-DOS screen or keyboard I/O, does not use MS-DOS memory management, and can even bypass MS-DOS file I/O with new Windows-specific device drivers. Version 3.0 Enhanced-mode can handle 32-bit device drivers that are not limited by the infamous 640K MS-DOS barrier. These drivers talk through Windows to applications that are also not limited by the constraints of MS-DOS.

The advantage of being able to work with MS-DOS is that it preserves the value added by MS-DOS's long life span (in computer years). Windows can run with MS-DOS TSRs, MS-DOS device drivers, and of course, it can run MS-DOS applications. Future versions of Windows will continue to be available on MS-DOS.

The Windows Architecture

Since the IBM PC was introduced in 1981, personal computers have become much more diverse in capability and in configuration. This diversity will increase in the next few years as personal computers based on RISC processors and multiprocessor systems are introduced.

These diverse systems have different operating system needs. For example, a low-end battery-operated portable requires minimal memory and hard disk footprint to minimize weight and cost. It also requires power management to extend battery life. In contrast, network servers and mission-critical desktops require sophisticated security to ensure the integrity of data. RISC-based systems require portability for both the operating system and the applications.

Some vendors feel that the diverse range of hardware requires totally different operating systems with incompatible applications. They sell different operating systems for personal computers, workstations, servers, and in the future, pen-based systems. Each of their operating systems require unique, incompatible applications. Connectivity between these divergent platforms is complicated.

Microsoft is focused on a much simpler solution. We're extending Windows into multiple, fully compatible implementations. Different implementations of Windows will be optimized for different classes of hardware. Customer investment in Windows development and Windows applications will be protected. Windows applications will run across the spectrum of hardware, from notepad-sized pen systems, to mission-critical desktops, to multiprocessor and RISC based systems.

Microsoft Windows is evolving into a complete operating system architecture. The Windows architecture addresses diverse requirements by supporting different modes of operation. Today Windows has three modes: Real-mode, Standard-mode, and Enhanced-mode. Real-mode provides compatibility with previous versions of Microsoft Windows. Standard-mode is optimized for an 80286 processor and provides access to the full 16 MB of memory supported by that chip. Enhanced-mode takes advantage of the 80386 and 80486 processors by providing support for multiple DOS applications and thru a technique called demand paging, provides applications with access to more memory then is physically present in the machine. All three modes support both DOS and Windows applications.

Building upon the success of Microsoft Windows Version 3.0, Microsoft will introduce Version 3.1 in late 1991. Version 3.1 incorporates significant customer feedback; it improves performance, introduces a newly designed file manager, improves network connectivity, and improves system reliability. Version 3.1 will support Windows Standard-mode and Enhanced-mode.

Also during 1991, Microsoft will enhance Windows Standard-mode and Enhanced-mode by providing extensions for sound, animation, and CD-ROM access, called Windows with Multimedia. We will also release extensions for clipboard and pen-style computing, called Microsoft Windows for Pen Computing.

WINDOWS NT

In 1992, Microsoft will introduce a new product called Windows NT (New Technology). Windows NT is built on a modern, 32-bit, operating system kernel. Windows NT will deliver an extremely robust client environment for mission-critical applications, a high-end desktop platform, and a portable, scalable server environment (See Figure 3). Windows NT will also transform Windows into a Microsoft LAN Manager server platform, thus adding a fourth server platform to the three currently supported by LAN Manager: OS/2®, UNIX, and VMS®.

Windows NT does not require DOS to function. However, it is compatible with the large installed base of DOS and Windows applications. In addition to compatibility with existing applications, Windows NT includes the features required to meet the needs of the high-end

desktop and server marketplace in the 1990s and beyond.

To support large server applications, Windows NT provides completely symmetric multiprocessor support. With Windows NT, tasks are symmetrically distributed between processors on a per thread basis. This design provides maximum utilization of each processor in a multiprocessor system and simplifies the development of multiprocessor applications.

Security is required for network servers and many mission-critical applications. To meet this need, Windows NT has been designed as a secure operating system. Microsoft is working with the U.S. Government to certify Windows NT as C2-level secure. In addition, the internal design of Windows NT can be enhanced in future releases to B-level security.

Windows NT is highly portable. It is being developed concurrently on x86 and MIPS-based RISC platforms. MIPS-based computers will be available from more than 60 hardware manufacturers who are members of the ACE (Advanced Computing Environment) consortium. With Windows NT, existing DOS and Windows programs will run unchanged on MIPS-based computers.

In addition to these advanced capabilities, the kernel-based design of Windows NT can be thought of as a nucleus which is compatible with different operating system environments. The kernel design provides Windows NT compatibility with DOS and Windows applications. It also allows Windows NT to support the OS/2 and POSIX application program interfaces, both of which are under development at Microsoft. This design also allows Windows NT to support applications written to the new Windows 32-Bit application program interface.

Windows 32-Bit API

Developers and end-users have made enormous investments in Windows programming and Windows applications. Most of these applications have been developed to run on both the 16-bit 80286 processor and the 32-bit 80386 and 80486 systems. Although highly capable, programs written to the Windows 16-Bit API, which is supported by Windows 3.0, are constrained by the memory limits inherent in a 16-bit architecture. Code must be divided into segments which cannot exceed 64K (65,536 bytes). This makes programming more difficult. It also imposes a performance penalty on high-performance 80486 and RISC-based systems.

The success of Windows Version 3.0, made it clear that an easy migration path from existing 16-bit Windows applications to a 32-bit programming interface was required. Unfortunately, OS/2 Presentation Manager™ (PM) was not an appropriate path. Although similar in appearance from an end-user perspective, the programming details of Windows and PM are completely different. So different in fact that most software developers found it necessary to completely rewrite their application when going from Windows to Presentation Manager. This is a major reason why there are so few OS/2 PM applications available.

The Windows 32-Bit API (Windows 32) has been designed to make the transition from the Windows 16-Bit API (Windows 16) to 32-bit as easy as possible. Only minimal changes have been made to the syntax of the Windows 32 API. The API names are the same as Windows 16. The semantics are identical. The message order is identical. In fact, it is possible to keep a single source code base and compile that source code into both 16-bit and 32-bit programs (see Figure 4).

While the Windows 32 API is extremely compatible with the Windows-16 API, it also

contains significant new features. These features include preemptive multitasked processes that use separate address spaces, preemptive threads, semaphores, shared memory, named pipes, mailslots, and memory mapped file I/O. GDI (Graphics Device Interface) improvements include Bézier curves, paths, transforms, and a device-independent color model.

The Windows 32-Bit API is fully supported in both Windows Enhanced-mode and Windows NT-mode. The Windows 32-Bit API will first be available in the Windows NT product during 1992. It will be added to Windows Enhanced-mode in late 1992 or early 1993. Programs written to the Windows 32-Bit API will run binary compatibly on both Windows NT-mode and Windows Enhanced-mode. All Windows 32 features are supported by both Windows Enhanced-mode and Windows NT-mode, including preemptive multitasking. Windows 32 programs will be fully source compatible between x86 and MIPS processors. Software Developer Kits for the Windows 32-Bit API will be available in late 1991.

The following highlights some key features of the Windows 32-Bit API.

Kernel: The Base Operating System

The Windows 32-Bit API on both Window NT-mode and Windows Enhanced-mode provides preemptive thread-based multitasking. It also runs all Windows 32-Bit and MS-DOS applications in separate address spaces so that they cannot corrupt one another.

The Windows 32-Bit API is designed to be portable beyond the 80386 and 80486 processors and in particular to be portable to RISC architectures. All these processors have different features but have in common 32-bit addressing and paged virtual memory architectures. Paged virtual memory is more efficient to implement and executes faster than segmented virtual memory. Memory management in Windows 32-Bit is secure because the operating system places different memory objects in different pages of memory and allows an application to control access permissions (Read, Write, Read/Write, Execute, and so on) to memory objects.

Given a large 32-bit address space, the operating system can conveniently and efficiently optimize file I/O because processes treat the file as a very large memory object and randomly access that object. The operating system, through page faulting, can detect read access to a file and bring in that data. It can detect when a shared file is written to and then write out that data. With process-settable access permissions and sparse allocation of physical memory pages, processes can implement very efficient data access, even when access patterns are entirely unpredictable.

GDI Improvements: Béziers, Paths, Transforms, Device-Independent Color

GDI, the drawing API for Windows Versions 3.0 and 3.1, provides a useful device-independent drawing set for applications. As output devices have become more sophisticated, so have drawing needs; hence GDI has been improved.

Some Windows applications for Versions 3.0 and 3.1 have needed to implement high-level graphics functions using the low-level drawing primitives of the Windows environment. Although this capability has provided application vendors flexibility in extending the Windows GDI, it has not allowed them to take seamless advantage of advances in printer and display technology. Application developers have had to code their own algorithms for displaying graphics such as Bézier curves and paths. With the Windows 32-Bit API, developers can call new high-level graphics features that will take advantage of the built-in drawing capabilities in advanced output hardware. Under Windows 32, displaying Bézier

curves can be handled by the graphics engine or by output devices that have implemented Bézier optimizations.

The Windows 32 GDI is a complete and general-purpose drawing package. Bézier curves are a general-purpose curve primitive from which a straight line can also be derived. This function combined with the PolyBézier functionality makes it possible to draw any combination of continuous lines and curves.

Windows 32 adds a Path API. A BeginPath EndPath sequence “closes” the sequence of drawing primitives between Begin and End. Thus, a BeginPath, PolyPolyBézier, EndPath sequence makes it possible to draw an arbitrary number of different filled shapes.

The Windows 32 transform API maps the virtual two-dimensional surface on which you draw to the two-dimensional output surface. This API, combined with the TrueType font technology first available with Windows Version 3.1, makes it possible to draw truly device-independent graphics that the system can map to the display surface, including the rotation of bitmaps, fonts, and metafiles.

Windows 32 will also provide a device-independent color model. Computer monitors and color printers use different technology to render colors. Additive mixing is used by computer monitors (RGB or Red, Green, Blue), while subtractive mixing (CYMK or Cyan, Yellow, Magenta, Black) is used by color printers. Without device-independent color, the different approach used by monitors and printers can result in mismatched colors.

The Windowing System and System Classes

The Windows windowing system is called User. The most significant change to User is the desynchronization of the per-window message queue from the system message queue. This change prevents errant, looping applications that stop processing their messages from blocking the computer system’s entire user interface, thus making other applications unavailable.

With the addition of input queue time thresholds, the system can provide default handling for a looping or an otherwise nonresponsive process. From an end-user perspective, this means the they can do other things while one application is busy. For example, if a word processing program is busy printing a 100 page document, a user can minimize that application and begin working on a spreadsheet. This effectively minimizes the time the user waits with an “hourglass” on their screen.

The desynchronization of the message queue is completely compatible with the Windows Versions 3.0 and 3.1 message model. The message ordering is the same. If WM_xyz came after WM_abc, it still does. This compatibility is entirely necessary because in Windows 32 systems, existing Windows applications run on top of the Windows 32-Bit message system. The messages are simply copied from the 32-bit stack to the 16-bit stack and passed onto the application; therefore message order cannot change.

Networking Extensions

Each time the APIs are further standardized in a particular area, it becomes easier to write significant new applications. Because of the variety of networking layers, ranging from network card interfaces and protocol stacks to the wide array of network IPC mechanisms, networking is probably the most confusing interface for developers today. Windows 32 will include standard network APIs that can replace those that network providers have previously needed to supply. Windows 32 will expose driver-style interfaces similar to the WinNet APIs provided by Windows Version 3.0 so that third-party vendors can plug their network services

into the Windows open architecture.

Some of the new, 32-bit WinNet APIs being defined are file, print, named pipes, mailslots, server browsing and machine configuration. This means applications can rely on a consistent programming interface regardless of the underlying network. Even if a network is not present, the APIs are still available and will return appropriate error codes.

The Windows 32-Bit API includes peer-to-peer named pipes, mailslots, and APIs to enable RPC (Remote Procedure Call) compilers. With Windows 32, a mail-server vendor can build a messaging service on named pipes and asynchrononous communication that will run on top of any network operating system, protocol stack, or network card — each of which could come from a different network vendor.

Compatibility with Windows 16-Bit APIs

Windows Version 3.0 and 3.1 applications will be able to run in Windows Enhanced-mode and Windows NT-mode systems that support the Windows 32 API. To be compatible with versions 3.0 and 3.1, all Windows 16 applications will run as one process in one address space. They will be nonpreemptive with respect to one another but preemptive with respect to the rest of the system, which mirrors their behavior under Windows Version 3.0 Enhanced-mode. Windows 16 applications run against the Windows 32-Bit APIs without a “layer” and without any state mapping or message reordering like that necessary to run them on OS/2 version 2.0.

Windows executables will also run on RISC-based Windows NT machines (see Figure 5). Excellent performance is expected on this platform because although some code will be run against 80286 emulator technology, all Windows calls will be mapped directly to the Windows NT software.

Windows 32-bit APIs—The Future of Windows

Millions of people are actively using Windows Version 3.0 today. Corporations and independent software vendors are making major commitments to Windows and investments in Windows applications.

To protect this investment, Microsoft is evolving Windows into a complete architecture. Through separate implementations, Microsoft Windows will run on vastly different types of hardware; from pen-based notepad computers to multiprocessor and RISC systems.

Windows NT and future versions of Windows Enhanced-mode will support the Windows 32-Bit APIs. Designed to simplify migration of Windows applications from 16-bit to 32-bit, these APIs will also make it easy to develop new 32-bit Windows applications. They contain significant new features which will enable a new generation of powerful Windows applications.

In addition, the Windows 32-Bit APIs will be used as the foundation for future versions of Windows under development at Microsoft. This technology, often called *Information at Your Fingertips*™*,* will make it even easier to use personal computers and will provide significant new functionality to Windows users.