
Vireo White Paper

Windows Device Driver Architectures

Stephen Lewin-Berlin
Vireo Software, Inc.

December 1997

Vireo Software presents articles of interest to the development community in the form of Vireo White Papers. For more information about the White Paper series, to request permission to reprint material, or to inquire about submitting an article to be published as a Vireo White Paper, please contact Vireo Software.

Vireo Software, Inc.
30 Monument Square
Suite 135
Concord, MA 01742

Phone: 978-369-3380
Fax: 978-318-6946

Email: Vireo@vireo.com
www.vireo.com

Windows Device Driver Architectures

Stephen Lewin-Berlin

The author is the president of Vireo Software, Inc. He can be reached via email at Berlin@vireo.com.

Vireo Software, Inc.
30 Monument Square
Suite 135
Concord, MA 01742

Phone: 978-369-3380
Fax: 978-318-6946

Email: Vireo@vireo.com
www.vireo.com

Table of Contents

Introduction

Microsoft offers a number of different operating systems that share the "Windows" moniker. Some underlying technology is shared across platforms, while other technologies are completely different. With the standardization of the Win32 API, Microsoft has encouraged compatibility of applications across various Windows platforms. Because device drivers interact with the lowest layers of the operating system, they must be compatible with the low-level architecture.

The underlying architecture of Windows 3.0 has remained constant through the Windows 3.x and Windows 9x families. Although the architecture has evolved and the number of interfaces available to device driver developers has expanded greatly, the underlying architecture has not changed. This allows device drivers written for Windows 3.x to continue to work, in most cases unmodified, on Windows 95 and Windows 98 platforms.

Windows NT was designed with a more modern and modular internal architecture. The benefits of the new architecture are indisputable in terms of flexibility and robustness. However, compatibility with Windows 3.x and Windows 9x was not a design goal, and is not provided.

Windows 98 introduces a "hybrid" internal architecture. Recognizing the value of compatibility across operating systems, Microsoft has attempted to unify the driver architecture in order to provide a single platform for future device driver development. Rather than introduce yet another new architecture, Microsoft has chosen to provide a design which is based on the more modern Windows NT architecture. The new device driver architecture is called the Win32 Driver Model, or WDM. The WDM architecture, while based on the NT kernel mode interface, does not support certain aspects of the NT model, such as the security model. WDM also incorporates plug and play features not available in Windows NT 4.0.

The term "WDM" is used to describe both a new driver API and a set of "class drivers" which allow the development of layered drivers for many interfaces such as USB devices. Windows 98 supports drivers written with either the Windows 95 (VxD) model or the new WDM model. Note that certain types of drivers for Windows 98 must be written as VxDs (such as file system drivers). Other types must be written as WDM drivers (such as USB drivers). In other cases, the developer may choose whether to write a VxD or WDM-style driver. When a choice is available, the developer may want to consider whether compatibility with Windows 95 or Windows NT is more important.

This paper discusses the basic differences between VxD, WDM, and NT Kernel Mode driver architectures, and presents guidelines that can be used to determine which type of device driver is best suited for particular applications. The paper concludes with a description of Vireo's device driver toolkits for the various Windows platforms.

A full and detailed description of the various architectures of Windows device drivers is beyond the scope of this paper. The bibliography lists several books and articles that include thousands of pages of detailed reference material.

VxD Architecture

VxDs are used in Windows 3.x, Windows 95, and Windows 98

What is a VxD?

The term “Virtual Device Driver” or VxD[1] describes a broad range of software that is used to extend the Windows operating system. The virtual device driver, originally designed to support hardware under Microsoft Windows, is a general purpose DLL that is linked into the most privileged part (ring 0) of the operating system. VxDs solve problems that cannot be solved by regular application programs. VxDs were introduced in the Windows 386 product, and the underlying architecture is used for Windows 3.x, Windows 95, and Windows 98.

The Virtual Machine Manager, or VMM, together with a confederation of VxDs shipped as part of the operating system, comprise the kernel of the Windows 3.x/9x system. User written drivers can call on some 900+ services to manage memory, access hardware, receive and dispatch interrupts, create network protocol stacks, implement file system components, and much more.

For a detailed description of the VxD architecture, refer to [VIREO], [HAZZAH], [ONEY1], or [DDK95].

When should I write a VxD?

If you need to write a low-level driver for Windows 3.x or Windows 95, you probably need to write a VxD. Rather than enumerate all of the kinds of drivers that are written as VxDs, it is simpler to list the kinds of device drivers that are NOT written as VxDs.

Printer drivers run at user privilege level, and are not VxDs.

DOS drivers can be installed before Windows runs. These drivers are not VxDs.

Microsoft has defined a miniport architecture for certain kinds of drivers. While these drivers call VxD services in order to interface to the operating system, the drivers themselves are not strictly VxDs. Rather, the drivers are PE files which are loaded by the PELDR component of the system. These files are binary compatible with Windows NT.

NDIS and SCSI drivers can be written as miniport drivers for Windows 95.

All other kinds of drivers for Windows 3.x and Windows 95 should be written as VxDs.

Drivers for Windows 3.1

Device drivers (VxDs) written for Windows 3.1 still work on Windows 95, and are expected to work on Windows 98 systems. These drivers were historically written in Intel assembly language until Vireo’s VtoolsD popularized the idea of using C or C++.

---

[1] VxD stands for Virtual ‘x’ Device. VxDs distributed by Microsoft have names such as VKD (Virtual Keyboard Device) and VDMAD (Virtual DMA Device).

Drivers for Windows 95

Windows 95 introduced quite a number of significant enhancements at the device driver level, many of which were designed to support Plug and Play. Among these enhancements are the system Registry, Pageable device drivers, and many services supporting dynamic driver loading.

The Windows 95 DDK also documented several significant enhancements that were introduced quietly in Windows for Workgroups, including the protected mode file system (IFS) and new communication driver architecture (VCOMM).

NT Kernel Mode Architecture

Kernel Mode Drivers are used in all versions of Windows NT

Device drivers written for Windows NT are traditionally written in C, and make calls to a Hardware Access Layer (HAL) when they need to gain direct control of the machine. The HAL provides an abstract layer between the driver and the actual hardware, and allows drivers to be source-compatible across different processors (such as Pentium and Alpha) by isolating hardware differences from the driver developer.

Because Windows NT runs on both single and multiple processor configurations, drivers written for Windows NT must take care to protect critical data structures in the case of multiprocessor systems.

NT provides a layered architecture for device drivers. An NT device driver has a lower and upper edge interface. Low-level drivers control hardware directly. *Low-level* drivers that expose an application interface (rather than being controlled by higher-level drivers) are known as *monolithic* drivers. Drivers that are layered above low-level drivers are known as *intermediate* drivers.

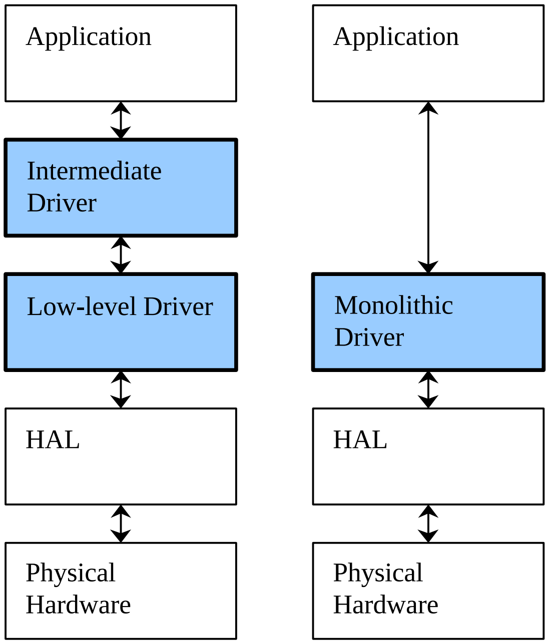

Figure 1 - NT layered driver model

Many device drivers are shipped as components of the Windows NT operating system. The interactions between the system-provided drivers define a layered architecture that third party developers can leverage. For example, there is a standard layering for SCSI and for NDIS drivers.

NT also defines a class driver architecture that supports certain classes of devices. For examples, there are SCSI class drivers to support SCSI tape drivers and SCSI disk drives. The architecture of class, port, and miniport drivers that is defined for Windows NT is extended for Windows 98 and Windows NT 5.0 in the Win32 Driver Model architecture.

Win32 Driver Model (WDM)

WDM drivers are planned for Windows 98 and Windows NT 5.0

The goal of Microsoft's WDM is to provide a common interface for device drivers across future Windows NT and Windows 98 systems. The current version of WDM is intended to be used for very specific classes of drivers such as "human input devices" (HID) and Universal Serial Bus devices (USB). It is expected that additional classes of drivers will be supported in future releases.

Sometimes WDM is the term used to identify the NT-style interfaces that are common to Windows NT 5.0 and Windows 98. Other times, WDM is used to signify the class driver/ minidriver architecture that has been defined for a specific set of devices.

As of this writing, Microsoft has defined WDM class drivers for specific hardware buses (USB and 1394); human input devices (HID) such as keyboards, mice, and joysticks; Digital Video Disks (DVD), still and video imaging, digital audio, and stream devices.

Each of these class/minidriver architectures is defined by a standard class driver provided by Microsoft, and an interface specification that describes the requirements for writing a minidriver of the appropriate type.

The WDM interfaces overlap with the Windows NT 4.0 device driver interfaces. WDM does not include [fred, help me here], but adds plug and play support and power management facilities.

How to choose between VxD, WDM, and Kernel Mode

The choice between the various architectures depends on the type of device driver you need to write, and the target platform or platforms.

- If you need a driver for Windows 95, write a VxD.
- If you need a driver for Windows NT 4.0, write an NT kernel mode driver.
- If there is a WDM class driver for your device, and your driver does not need to be compatible with Windows 95 or Windows NT 4.0, then write a WDM driver.
- If your primary target is Windows 98, then you must decide whether you prefer to write a driver that is more compatible with Windows 95 or with Windows NT.

  If you prefer Windows 95 compatibility, then you should write a VxD.

  If you prefer Windows NT compatibility, then write a WDM driver. Note that the WDM driver will not operate on Windows 95 or Windows NT 4.0. However, the WDM driver will use the same general structure as a driver for Windows NT. This makes it easier to build a variant of the driver for Windows NT.

- If your primary target is Windows NT version 5.0, then you must decide whether you prefer to write a driver that is more compatible with Windows NT 4.0 or with Windows 98.

  If you prefer Windows NT 4.0 compatibility, then you should write an NT kernel mode driver.

  If you prefer Windows 98 compatibility, then write a WDM driver. Note that the WDM driver will not operate on Windows NT 4.0. However, the WDM driver will use the same general structure as a driver for Windows NT 4.0. This makes it easier to build a variant of the driver for Windows NT 4.0.

Vireo Device Driver Development Toolkits

Vireo provides device driver toolkits for Windows 3.1, Windows 95, Windows 98, and all version of Windows NT.

If you need to write a VxD, we recommend using VtoolsD. The VtoolsD toolkit includes everything you need to build VxDs with a compiler from either Microsoft or Borland. Note that the Microsoft DDK is NOT required if you are using VtoolsD for Windows 95.

VtoolsD will include complete support for VxD development on Windows 3.1, Windows 95, and Windows 98. Support for Windows 98 is subject to change until a final release of the DDK is released by Microsoft.

If you need to write an NT driver, we recommend using Driver::Works. The Driver::Works toolkit greatly simplifies the development process through the use of a carefully designed class library, examples, and a code generation wizard that integrates into the Microsoft Visual C++ environment.

Driver::Works also includes support for WDM development on either Windows 98 or Windows NT 5.0 platforms. The WDM support in Driver::Works is subject to change until the corresponding operating systems are released. Vireo will issue periodic updates to ensure that the tools are synchronized with the latest beta releases of the Microsoft DDK and the beta releases of Windows 98 and Windows NT 5.0.

For more information about Vireo’s device driver solutions, visit the Vireo web site at http://www.vireo.com.

Vireo Software, Inc.
30 Monument Square
Suite 135
Concord, MA 01742

Phone: 978-369-3380
Fax: 978-318-6946

Email: Vireo@vireo.com
www.vireo.com

References

[VIREO] Vireo Software. VTOOLSD User's Guide. Concord, MA, Vireo Software, 1994-1997.

[VIREO2] Vireo Software. Driver::Works User's Guide. Concord, MA, Vireo Software, 1997.

[HAZZAH] Karen Hazzah. Writing Windows VxDs and Device Drivers. Lawrence, KS, R&D Publications, 1995-1997.

[ONEY1] Walter Oney. Systems Programming for Windows 95. Redmond, WA, Microsoft Press, 1996.

[DDK95] Microsoft Corporation. Windows 95 Device Driver Kit. Redmond, WA, 1991-1997.

[DDKNT] Microsoft Corporation. Windows NT Device Driver Kit. Redmond, WA, 1991-1997.

[BAKER] Art Baker. The Windows NT Device Driver Book. Upper Saddle River, NJ, 1997.

[ONEY2] Walter Oney, "Surveying the new Win32 Driver Model for Windows 98 and Windows NT 5.0," *Microsoft Systems Journal*, November 1997, Vol. 12, No. 11.

[ONEY3] Walter Oney, "Implementing the new Win32 Driver Model for Windows 98 and Windows NT 5.0," *Microsoft Systems Journal*, December 1997, Vol. 12, No. 12.

Vireo Software

Vireo Software, Inc.
30 Monument Square
Suite 135
Concord, MA 01742

Phone 978-369-3380
Fax 978-318-6946

Vireo@vireo.com
www.vireo.com