Figures

Overview

A significant new feature of IBM* OS/2* Version 2.0 is the ability to execute multiple DOS applications concurrently, with pre-emptive multitasking and full memory protection for each application. Microsoft** Windows** applications are also supported in the same way. These capabilities allow the use of OS/2 Version 2.0 as an integration platform for DOS applications, Windows applications, and OS/2 applications in a seamless, fully functional environment.

This chapter provides a brief overview of the OS/2 Version 2.0 product and the Multiple Virtual DOS Machines component which provides support for DOS and Windows applications. DOS and Windows application support is then described in more detail in the remainder of the document.

MVDM Architecture

The Multiple Virtual DOS Machines component of OS/2 Version 2.0 is itself comprised of a number of subcomponents, which interact with one another and with the OS/2 Version 2.0 operating system kernel to provide services to DOS applications running in virtual DOS machines. This chapter describes each subcomponent of MVDM, and explains the interaction between subcomponents.

8086 Emulation

A significant capability of the Intel 80386 and 80486 processor families is their ability to emulate the Intel 8086 processor. This emulated state is known as **virtual 8086 (V86) mode**. The Multiple Virtual DOS Machines component of OS/2 Version 2.0 makes use of this mode to provide emulation of a native DOS environment for applications executing in virtual DOS machines. The 8086 processor hardware emulation is provided by the 8086 Emulation component of MVDM. Other aspects of the DOS environment, such as program loading and execution and device driver support, are provided by the DOS Emulation component, which is described in MVDM DOS Emulation, and by virtual device drivers, described in Device Drivers.

This chapter provides an overview of V86 mode, and describes the operation of the 8086 Emulation component.

MVDM DOS Emulation

MVDM DOS Emulation provides DOS services to applications running in Multiple Virtual DOS Machines. The environment provided is highly compatible with those same services that are provided to DOS applications running under native DOS 5.0. It addresses only DOS compatibility aspects; processor and other hardware aspects of the DOS environment are addressed by the 8086 Emulation component and virtual device drivers.

This chapter describes the implementation of the DOS Emulation component, and the DOS software services provided to support DOS applications running in virtual DOS machines.

Device Drivers

In order to provide the maximum level of hardware independence for the OS/2 Version 2.0 operating system, device driver are used to communicate with hardware devices. This concept is not new, and has been implemented in previous versions of OS/2 and in the DOS operating system. However, the implementation of device drivers under OS/2 Version 2.0 differ in several significant ways from that seen in previous versions. This chapter describes the implementation of device drivers under OS/2 Version 2.0.

Memory Extender Support

Many popular DOS applications use memory extenders such as EMS and/or XMS to gain access to memory above the 1MB real mode addressing limit on the 80286, 80386, or 80486 processors. Such extenders allow DOS applications to have total code and data spaces larger than the available base memory, and to have very large code or data objects loaded into memory for enhanced execution speed. The standard configuration of OS/2 Version 2.0 provides both LIM EMS Version 4.0 (which includes backward compatibility with LIM Version 3.X) and LIMA XMS Version 2.0 functions for DOS applications running in virtual DOS machines.

Figure "General Overview of Different Types of Memory for DOS Applications"

This chapter describes the implementation of EMS and XMS support for virtual DOS machines. For those readers not already familiar with the architecture of these memory extenders, an overview is provided in Memory Extender Architectures.

DOS Protected Mode Interface

Perhaps the most significant limitation of real mode operation, as used by DOS and similar operating systems, is the 1MB addressing limitation. This limitation can be overcome by executing applications in protected mode, but since the DOS operating system and most TSR applications must run in real mode, problems arise when applications attempt to access interrupts, TSRs or operating system facilities.

The **DOS Protected Mode Interface** (**DPMI**) is a protected mode programming interface for DOS applications, allowing such applications to run on an 80286 or 80386 processor in protected mode, while utilizing the real mode services of the operating system and device drivers. When an application wishes to access a DOS service, it makes a request to DPMI, which handles the appropriate address translations, switches the processor to real mode and makes the service request to DOS. The result of the request is then translated to the correct format for the protected mode application, the processor is switched back to protected mode, and control is returned to the application.

DPMI has been implemented in the Multiple Virtual DOS Machines component of OS/2 Version 2.0, and provides functions such as memory allocation and interrupt management for applications which use DPMI services. This support is provided through a component, implemented as a virtual device driver, known as the **DPMI API Layer**, in conjunction with the MVDM kernel.

This chapter provides a brief overview of DPMI, and describes its implementation under OS/2 Version 2.0.

Running DOS Applications

OS/2 Version 2.0 allows the user to run multiple DOS applications concurrently, with each application running in its own virtual DOS machine, with pre-emptive multitasking and full memory protection.

This chapter describes the way in which a DOS application can be defined to the OS/2 Version 2.0 Workplace Shell, and the ways in which an application may be started. It also discusses the way in which version-specific DOS applications may be run in virtual DOS machines under OS/2 Version 2.0.

Virtual Machine Boot

An important goal of OS/2 Version 2.0 is the ability to run past, current, and future DOS programs; indeed most DOS applications available today run unchanged in the MVDM environment.

However, it should be remembered that the "DOS" which runs in this case is highly optimized for (and specific to) an OS/2 Version 2.0 virtual 8086 machine. Because of this, there are fundamental internal differences between the DOS Emulation provided under OS/2 Version 2.0 and "real" DOS. Two problems may therefore arise:

1. Some programs may depend on specific DOS features such as internal control blocks, LAN Redirector hooks, or even undocumented functions, which are not present in MVDM's DOS Emulation.
2. Only DOS *character* device drivers can be loaded in MVDM's DOS Emulation. The user may own a *block* device (such as a special disk or tape drive) for which no OS/2 driver is available. (A block device is one accessed via a drive letter, such as E:).

Virtual Machine Boot (**VMB**) solves both these problems. It allows the user to boot "off-the-shelf" DOS and use block device drivers in an OS/2 Version 2.0 virtual DOS machine, ensuring greater compatibility for DOS applications.

Another possible use of VMB is to run DOS of a different National Language to that of OS/2 Version 2.0 itself. This may be useful in a multilingual environment.

Memory Extender Architectures

This appendix provides an overview of the LIM EMS Version 4.0 and LIMAXMS memory extender architectures, for those readers who require an understanding of their implementation and who are not already familiar with the design of these memory extenders.

Multiple Virtual DOS Machines Lab Sessions

These lab sessions provide practical demonstrations of OS/2 Version 2.0's Multiple Virtual DOS Machine capabilities. The individual topics that will be covered in this lab are:

- VDM Configuration
- The Virtual DOS Machine Manager
- Using the Clipboard
- VDM Use of the Speaker
- VDM Interprocess Communications