IBM

International Technical Support Centers

OS/2 2.1 Technical Update

GG24-3948-00

OS/2 2.1 Technical Update

Document Number GG24-3948-00

May 1993

International Technical Support Center
Boca Raton

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xvii.

First Edition (May 1993)

This edition describes the new features and functions of OS/2 2.1.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSC Technical Bulletin Evaluation Form for readers' feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Center
Dept. 91J, Building 235-2 Internal Zip 4423
901 NW 51st Street
Boca Raton, Florida 33431-1328

Abstract

This document describes OS/2 2.1. It provides an overview of all the new and changed functions in OS/2 2.1, and a detailed technical discussion of the major new topics.

It also includes advice on programming Workplace Shell objects which is applicable generally to OS/2 Version 2, as well as information on OS/2 2.00.1 and the Service Pak XR06055 for OS/2 2.0.

This document should be used in conjunction with the *OS/2 Version 2.0 Technical Compendium*, GBOF-2254.

This document is intended for IBM customers, dealers, system engineers, consultants and others who need to know what is new in OS/2 2.1 and to understand the benefits. A knowledge of OS/2 2.0 is assumed.

PS (363 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM customers, dealers, system engineers and consultants to understand the new and changed features in OS/2 2.1. It also includes advice on programming Workplace Shell objects which is applicable generally to OS/2 Version 2, as well as information on OS/2 2.00.1 and the Service Pak XR06055 for OS/2 2.0. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/2 2.1. See the PUBLICATIONS section of the IBM Programming Announcement for OS/2 2.0 and OS/2 2.1 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

ActionMedia
AS/400
Audio Visual Connection
IBM
M-Motion
Micro Channel
Multimedia Presentation Manager/2
OS/2
Presentation Manager
PS/1
PS/2
PS/ValuePoint
ThinkPad
Trackpoint II
Ultimedia
Ultimotion
WIN-OS/2

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

386, 486, SX, Pentium and Indeo are trademarks of Intel Corporation.
1-2-3 and Lotus are trademarks of the Lotus Development Corporation.
ActionMedia II and Audio Visual Connection are registered trademarks of Digital Video Interactive.
Adobe Type Manager is a registered trademark of Adobe Systems, Inc.
Amdek and SmartVision are trademarks of Wyse Technology.
ATI is a trademark of ATI Technologies, Inc.
AutoCad is a trademark of AutoDesk, Inc.
Boca Research is a trademark of Boca Research, Inc.
CompuServe is a registered trademark of CompuServe, Inc.
Consulting Detective is a trademark of Sleuth Productions Ltd., licensed to ICON Simulations Inc.
ERGO, STB and Powergraph are trademarks of STB Systems, Inc.
Everex and Viewport are trademarks of Everex Systems, Inc.
Hitachi is a trademark of Hitachi Corporation.
Kodak is a trademark of Eastman Kodak Company.
Legend is a trademark of Sigma Designs Inc.
NEC is a trademark of NEC Corporation.
Orchid and ProDesigner are trademarks of Orchid Technology, Inc.
Omnipage Professional is a trademark of Caere Corporation.
Panasonic is a trademark of Matsushita Electric Industrial Co., Ltd.
PCMCIA is a trademark of the Personal Computer Memory Card International Association.
Pioneer is a trademark of Pioneer Electronic Corporation.
PostScript is a trademark of Adobe Systems, Inc.
Pro Audio Spectrum is a trademark of Media Vision.
Sony is a trademark of Sony Corporation.
SoundBlaster is a trademark of Creative Labs, Inc.
Speedstar is a trademark of Diamond Computer Systems, Inc.
Toshiba is a registered trademark of Toshiba Corporation.

TrueType is a registered trademark of Apple Computer Inc.
Tseng is a trademark of Tseng Laboratories, Inc.
VGAWONDER XL is a trademark of ATI Technologies Inc.
Microsoft Windows, Microsoft and Excel are trademarks of Microsoft Corporation.

Preface

This document describes OS/2 2.1. It provides an overview of all the new and changed functions in OS/2 2.1, and a detailed technical discussion of the major new topics.

Additional advice is also provided on programming Workplace Shell objects, which is applicable generally to OS/2 Version 2.

It also includes information on OS/2 2.00.1 and the Service Pak XR06055 for OS/2 2.0, and positions these interim releases of OS/2 Version 2 relative to OS/2 2.0 and OS/2 2.1.

This document was written for persons requiring an overview and technical description of the new and changed functions in OS/2 2.1, and also of the interim releases of OS/2 2.00.1 and OS/2 2.0 with Service Pak XR06055 applied.

This document is organized as follows:

- Chapter 1, "Overview of OS/2 2.1"

  This chapter provides an introduction to the OS/2 2.1 features, packaging and upgrade options.

- Chapter 2, "Summary of OS/2 2.1 Enhancements"

  This chapter provides a detailed summary of the enhancements contained in OS/2 2.1, from both a user and a system perspective. These enhancements are described in more detail in the succeeding chapters. It also contains a discussion of the performance improvements in OS/2 2.1 for 32-bit OS/2 and 16-bit Windows applications.

- Chapter 3, "Installation and Configuration of OS/2 2.1"

  This chapter provides advice, based on experience, on the installation and configuration of OS/2 2.1.

- Chapter 4, "Hardware Support and Enhancements"

  This chapter describes the additional hardware support provided in OS/2 2.1. It includes discussions of system, disk, CD, and video device support.

- Chapter 5, "Control Program Enhancements"

  This chapter describes the enhancements to the OS/2 2.1 Control Program.

- Chapter 6, "MVDM Enhancements"

  This chapter discusses the enhancements to the DOS MVDM support, include the dual-thread enhancement which benefits DOS multimedia applications.

- Chapter 7, "WIN-OS/2 3.1 Support and Enhancements"

  This chapter describes the major improvements in WIN-OS/2 capability in OS/2 2.1, which now includes support for Windows 3.1 applications as well as performance and usability enhancements.

- Chapter 8, "Presentation Manager Enhancements"

  This chapter provides a description of the new 32-bit Graphics Engine, along with a discussion of the new 32-bit Presentation Manager display drivers for XGA/XGA-2, SVGA, 8514/A and VGA.

- Chapter 9, "Workplace Shell Enhancements"

  This chapter describes the enhancements to the Workplace Shell, including improved INI file handling, and minor visual and drag/drop enhancements.

- Chapter 10, "Print Subsystem Enhancements"

  This chapter describes the enhancements to the Print Subsystem, including improved printer driver installation, as well as improved and additional printer device drivers.

- Chapter 11, "Enhancements for Laptops and Notebooks"

  This chapter describes the enhancements in OS/2 2.1 which provide improved support for mobile computing, such as Advanced Power Management and PCMCIA support.

- Chapter 12, "MMPM/2 - The OS/2 Multimedia Environment"

  This chapter discusses the multimedia support in MMPM/2 1.1 which is now packaged as standard with OS/2 2.1.

- Chapter 13, "Tuning and Performance Characteristics"

  This chapter includes advice and guidance for tuning and improving the performance of an OS/2 Version 2 system.

- Chapter 14, "Workplace Shell Programming"

  This chapter provides additional advice on developing a SOM-based Workplace Shell application under OS/2 Version 2. It is an expanded version of Chapter 7 in *OS/2 Version 2.0 - Volume 4: Application Development*, and includes information on implementing features such as drag-and-drop, and advice on debugging Workplace Shell applications.

- Appendix A, "OS/2 2.00.1 and Service Pak XR06055 Enhancements"

  This chapter describes OS/2 2.00.1 and the Service Pak XR06055 for OS/2 2.0, and compares them to OS/2 2.0 and OS/2 2.1. It discusses upgrading these interim releases to OS/2 2.1, and also any special considerations for use and installation.

- Appendix B, "OS/2 2.x Compatible PCM Systems"

  This appendix contains a list of the PCs from various PC manufacturers on which OS/2 2.x has been tested.

- Appendix C, "OS/2 2.x Compatible Hardware Devices"

  This appendix contains a list of the adapters and peripherals from various hardware manufacturers on which OS/2 2.x has been tested.

- Appendix D, "APAR Fixes in OS/2 2.00.1, Service Pak XR06055, and OS/2 2.1"

  This appendix provides a summary list of the APAR fixes in OS/2 2.00.1, the Service Pak XR06055 for OS/2 2.0, and OS/2 2.1.

- Appendix E, "OS/2 2.0 Volumes 1-5: Clarifications and Corrections"

  This appendix contains a list of the clarifications and corrections to *Volumes 1 through 5 of the OS/2 Technical Compendium.*

Related Publications

The following publications are considered particularly suitable for a more detailed discussion of the topics covered in this document.

International Technical Support Center Publications

- *OS/2 Version 2.0 Technical Compendium*, GBOF-2254, which comprises:
  - *OS/2 Version 2.0 - Volume 1: Control Program*, GG24-3730
  - *OS/2 Version 2.0 - Volume 2: DOS and Windows Environment*, GG24-3731
  - *OS/2 Version 2.0 - Volume 3: Presentation Manager and Workplace Shell*, GG24-3732
  - *OS/2 Version 2.0 - Volume 4: Application Development*, GG24-3774
  - *OS/2 Version 2.0 - Volume 5: Print Subsystem*, GG24-3775
- *OS/2 Version 2.0 Remote Installation and Maintenance*, GG24-3780-01
- *Automated Installation for CID Enabled OS/2 V2.0*, GG24-3783
- *Automated Installation for CID Enabled Extended Services, LAN Server V3.0 and NTS/2*, GG24-3781
- *Multimedia Presentation Manager/2 Programming Using Digital Audio Examples*, GG24-3963
- Multimedia Presentation Manager/2 Advanced Programming Techniques, GG24-4042
- *IBM PS/2 and PS/ValuePoint Subsystems*, GG24-4002
- *Advanced PS/2 Servers - Planning and Selection Guide*, GG24-3927

A complete list of International Technical Support Center publications, with a brief description of each, may be found in:

- *Bibliography of International Technical Support Centers Technical Bulletins*, GG24-3070.

Publications Shipped with OS/2 2.1

- *OS/2 2.1 Using the Operating System*
- *OS/2 2.1 Installation Guide*

Both of these documents can also be ordered as part of the OS/2 2.1 documentation-only package, S61G-0905.

Other Publications

- *The Design of OS/2 (by Harvey Deitel and Michael Kogan)*, S325-4005-00 and ISBN 0-201-54889-5
- *Client/Server Programming with OS/2 2.0, 2nd edition (by Robert Orfali and Dan Harkey)*, G325-0650-01 and ISBN 0-442-01219-5
- *OS/2 2.1 Unleashed (by David Moskowitz and David Kerr)*, SR28-4318-00 and ISBN 0-672-30240-3

- *OS/2 2.1 Integrationsplattform (by Dorle Hecker and Hans Goetz)* , ISBN 3-7723-4981-1. This is written in German.
- *Inside OS/2 2 Special Edition (by Mark Minasi, John Little, Marlene Semple and Bill Camarda)*, G362-0016-01 and ISBN 1-56205-134-2
- *Maximizing OS/2 (by John Little, Toby Pennycuff, Marlene Semple and Stephen Gutknecht)*, SR28-4320 and ISBN 1-56205-118-0
- *IBM Personal Systems Technical Solutions (published by IBM PS Competency Center, Roanoke, Texas)*, including:
  - *April 1992 issue*, G325-5015-00
  - *July 1992 issue*, G325-5017-00
  - *October 1992 issue*, G325-5019-00 (especially the article on "OS/2 Installation and Performance Considerations")
  - *January 1993 issue*, G325-5012-00 (especially the articles on "Loadable ABIOS" and "Trackpoint II")
  - *April 1993 issue*, G325-5021-00 (especially the article on "IBM Personal Software Products: Product Line Update")
- *OS/2 Developer (published by IBM PS Line of Business, Boca Raton, Florida)*, including:
  - *Summer 1992 issue*, G362-0001-14 (especially the articles on "Class Objects in SOM" and "The OS/2 Workplace Programming Interface)
  - *Fall 1992 issue*, G362-0001-15
  - *Winter 1993 issue*, G362-0001-16
  - *OS/2 2.X Notebook (a compendium of the above magazines)*, G362-0015-00 and ISBN 0-442-01522-4
- *OS/2 2.0 Technical Library* (published by IBM Personal Systems Programming, Boca Raton, Florida), including:
  - *Application Design Guide*, S10G-6260-00
  - *Programming Guide Volume I*, S10G-6261-00
  - *Programming Guide Volume II*, S10G-6494-00
  - *Programming Guide Volume III*, S10G-6495-00
  - *System Object Model Guide and Reference*, S10G-6309-00
  - *PM Programming Reference Volume II*, S10G-6265-00
  - *PM Programming Reference Volume III*, S10G-6272-00
- *IBM OS/2 Version 2.0 - Information and Planning Guide*, G326-0160-00
- *The OS/2 Multimedia Advantage*, S41G-2923
- *MMPM/2 1.1 Application Programming Guide*, S71G-2221
- *MMPM/2 1.1 Subsystem Development Guide*, S71G-2223
- *CUA Guide to Multimedia User Interface Design*, S41G-2922

CD-ROM Publications

- *IBM OS/2 Online Book Collection*, S53G-2166
- *IBM OS/2 Device Driver Development Kit*, DDK01-93

Acknowledgements

The project leader and editor for this project was:

Adam Jollans
IBM International Technical Support Center, Boca Raton

The authors of this document are:

Florence Calvez, IBM France
Rich Dulaney, IBM Personal Systems Programming Center, Boca Raton
Goh Song Eng, IBM Singapore
John Heggie, IBM Australia
Don Johnston, IBM ITSC Boca Raton
Adam Jollans, IBM ITSC Boca Raton
Dietmar Juretzka, IBM Germany
Thomas Löffler, IBM ITSC Boca Raton
Douglas Pearless, IBM New Zealand
David Saunders, IBM European Personal Systems Center, Basingstoke
Jerry Stegenga, IBM ITSC Boca Raton
Sean Whalen, IBM United States
Young Woo Pae, IBM Korea

This publication is the result of a series of residencies conducted at the International Technical Support Center, Boca Raton.

Thanks to the following people from the IBM Personal Systems Programming Center at Boca Raton, for their invaluable advice, guidance and support provided in the production of this document:

Ayo Anise
Craig Bennett
Arnold Bramnick
Glenn Brew
Ron Cadima
Joe Celi
Bob Chernow
Marc Cohen
Rick Efruss
Mark Fiechtner
Bruce Gobioff
Kathleen Hamill
Kip Harris
Wally Harris
Khoa Huynh
Ron Kenney
David Kerr
Dorothy Kruger
Salil Kulkani
Mary Jane de Laurentis
Fred Lathrop
Kelvin Lawrence
Joe Luu
Peter Magid

Dave Marshall
Jim Magnuson
Linda Magnuson
Patrick McCourt
Darren Miclette
Felix Miro
Pam Mitarotondo
Jeff Muir
Pat Nogay
Michael Perks
David Reich
Ginny Roarabaugh
Robbie Robinson
Pete Rodriguez
Felix Ruo
Shon Saliga
Laura Sanders
Frank Schroeder
Joe Sermarini
John Sierra
Greg Simco
Barry Straub
Vickie Szarek
James Taylor
Bernie Thompson
Michelle Vaghari
Diane Walewski
Ted Waldron
Franz Walkow
Dave Whiter
Steve Woodward
Mary Wright

Thanks especially to Linda Magnuson, the OS/2 2.1 planner at IBM Boca Raton, for her support and help with this project.

Thanks also to the following people from outside Boca for their detailed reviews and comments on drafts of this document:

Mike Kaply, IBM PSP, Austin
Martin McElroy, IBM PSP EMEA, Basingstoke
Michael Price, IBM UK, London

Portions of this document are based on *IBM Multimedia Presentation Manager/2 - Getting Started*, S41G-8306 and *The OS/2 Multimedia Advantage*, S41G-2923, and also on OS/2 Installation and Performance Considerations, published in *IBM Personal Systems Technical Solutions, October 1992*, G325-5019-00.

Appendix B, "OS/2 2.x Compatible PCM Systems" on page 305 and Appendix C, "OS/2 2.x Compatible Hardware Devices" on page 321 have been provided by IBM's Worldwide Industry Hardware Support group in the Personal Systems Programming laboratory at Boca Raton in Florida.

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Center
Department 91J, Building 235-2
Internal Zip 4423
901 NORTHWEST 51ST STREET
BOCA RATON FL
USA 33431-1328

Fold and Tape **Please do not staple** Fold and Tape

ITSC Technical Bulletin Evaluation RED000

OS/2 2.1 Technical Update

Publication No. GG24-3948-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) Are you an employee of IBM or its subsidiaries? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

Chapter 1. Overview of OS/2 2.1

The aim of **OS/2* 2.1** is to run a wide range of PC applications, reliably and productively, on a wide range of PC hardware.

It is based on **OS/2 2.0**, which provided an advanced 32-bit multitasking operating system for IBM* and IBM-compatible PCs, and the capability to run DOS, Windows** and OS/2 applications.

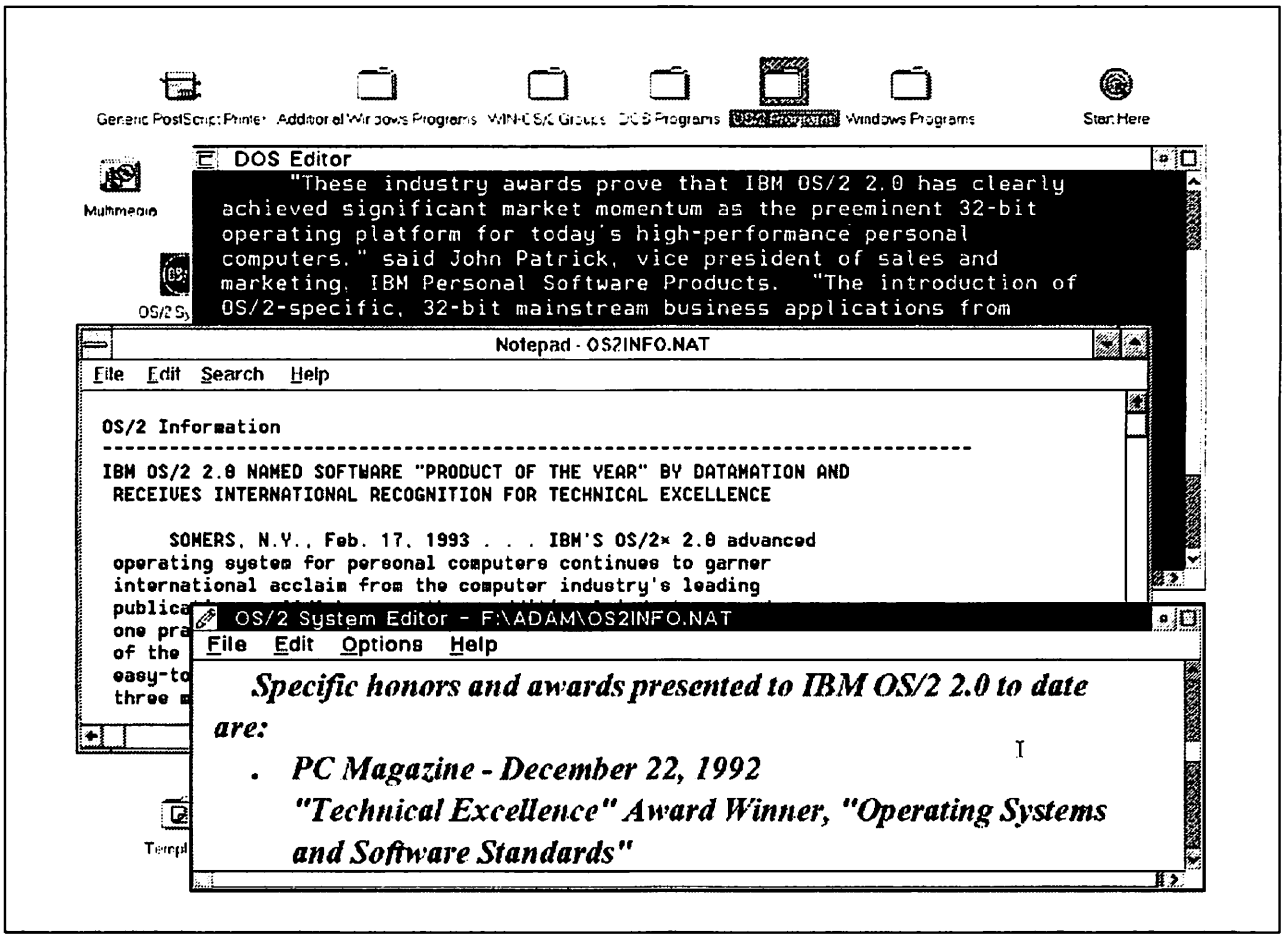


Figure 1. OS/2 2.1 Running DOS, Windows 3.1 and OS/2 Applications

OS/2 2.1 provides major enhancements in these three areas:

- **Improved Application Support**
  - Windows 3.1 applications (including multimedia), with WIN-OS/2* 3.1.
  - OS/2 multimedia applications, with Multimedia Presentation Manager/2* (MMPM/2) 1.1
- **Operating System Engine Upgrades**
  - 32-bit Presentation Manager* graphics engine
  - Performance enhancements, especially for the WIN-OS/2 environment
  - Substantial code optimization and defect removal since the release of OS/2 2.0
- **Wider Hardware Support**
  - Installation of OS/2 2.1 from CD-ROM

- OS/2 2.1 now preloaded by many PC manufacturers
- Additional SCSI adapter and CD-ROM drive support
- 32-bit seamless PM and WIN-OS/2 display drivers for XGA, XGA-2, 8514/A, VGA. and many SVGA adapters
- Additional printer support for both PM and WIN-OS/2
- APM and PCMCIA support for laptop and notebook computers

Some of these functions were introduced with the interim releases of OS/2 2.00.1 and the Service Pak XR06055 for OS/2 2.0. However, the majority of the features listed above are new with OS/2 2.1.

For a list describing which features are new in OS/2 2.1 and which were also in OS/2 2.00.1 and the Service Pak XR06055 for OS/2 2.0, please refer to Appendix A, "OS/2 2.00.1 and Service Pak XR06055 Enhancements" on page 289.

1.1 OS/2 2.1 Packaging

OS/2 2.1 is an upgrade to OS/2 2.0, and should be seen as an enhancement and replacement for OS/2 2.0.

Although it is possible to continue on OS/2 2.0 (with the Service Pak XR06055 for OS/2 2.0 applied) and apply further APAR fixes, OS/2 2.1 provides an enhanced and stable platform for the future, and we recommend all OS/2 2.0, OS/2 2.00.1 and OS/2 2.0 (with the Service Pak XR06055 for OS/2 2.0 applied) users to upgrade.

OS/2 2.1 is manufactured and sold in three different packages. All three packages contain the same enhancements and fixes.

Full Package

This package should be used by users who do not currently have OS/2 or DOS (with or without Windows) installed on their systems.

This package is sold and distributed in three formats:

- 3.5" diskettes
- 5.25" diskettes
- CD-ROM

The CD-ROM package also contains two 3.5" installation diskettes and two 5.25" installation diskettes, which are needed for the initial IPL of OS/2 2.1.

Upgrade Package

This package should be used by users who currently have either OS/2 or DOS (with or without Windows) installed. When installing this version, OS/2 2.1 will look to see which operating system is currently installed on the system. If no operating system is found, then OS/2 will not install.

The upgrade package is available in the same three formats as for the full package, that is:

- 3.5" diskettes
- 5.25" diskettes
- CD-ROM

Preload Package This package will be preloaded onto many systems by IBM and other PC manufacturers.

The preloaded system contains utilities for creating a set of bootable utility diskettes, uninstalling OS/2 features, and configuring the system setup. The utility diskettes which can be created are for supporting the preloaded system in case problems are encountered and the system cannot be booted normally.

Backup copies of the diskettes are supplied with some systems; the inclusion of backup diskettes varies between countries.

The upgrade from OS/2 2.0 to OS/2 2.1 is not free, and a license must be held for each copy of OS/2 2.1.

For the full and upgrade packages these licenses must be purchased. For the preloaded systems these licenses are included with the hardware.

Public beta-test versions of OS/2 2.1 were released during late 1992 and early 1993. The license agreement for these beta-test versions has expired with the general availability of OS/2 2.1, and any remaining copies of the beta-test versions should now be replaced by licensed copies of OS/2 2.1.

1.2 Upgrading to OS/2 2.1

OS/2 2.1 is available as a shrink-wrapped package, and can be installed either from diskette or from a combination of CD-ROM and two boot diskettes.

In addition, OS/2 2.1 is available as a diskette upgrade package from any of the previous versions of OS/2 Version 2 (OS/2 2.0, Preloaded OS/2 2.00.1 and OS/2 2.0 with Service Pak XR06055 applied), as shown in Figure 2 on page 4.

An OS/2 2.1 upgrade package is also available to move from OS/2 1.x or DOS (with or without Windows) to OS/2 2.1.

OS/2 2.1 is preloaded onto a wide range of systems by a variety of PC manufacturers. For more details, see section 4.11, "Preloaded IBM Hardware Systems" on page 60.

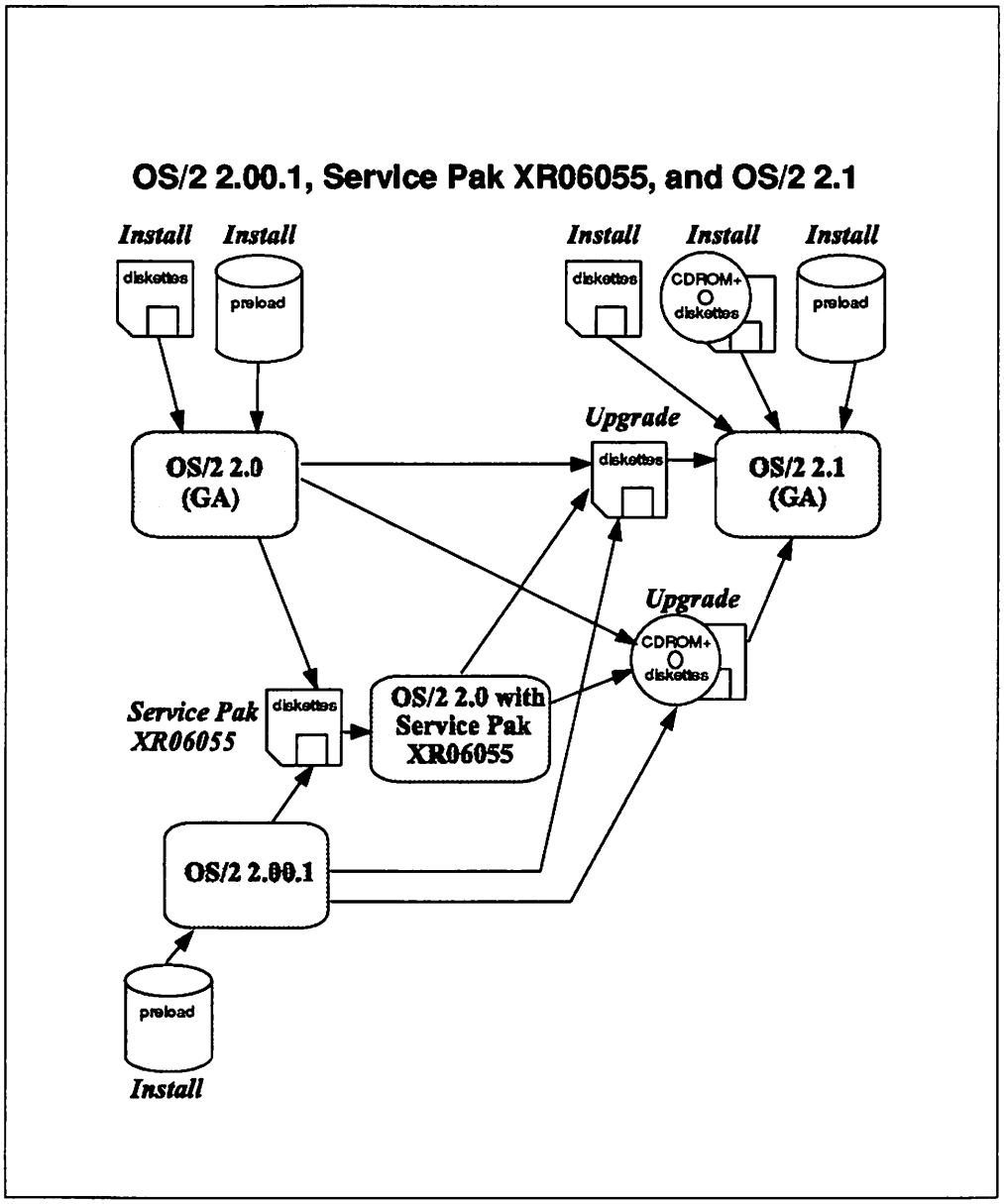


Figure 2. Upgrade Routes to OS/2 2.1

Chapter 2. Summary of OS/2 2.1 Enhancements

OS/2 2.1 was announced worldwide in May 1993, and provides the upgrade path from OS/2 2.0.

The enhancements included in OS/2 2.1 can be viewed from two complementary perspectives:

- **User Perspective**
  - What are the enhancements, in a rough order of importance?
  - Are these enhancements related to application support, internal changes to the operating system, or hardware support?
  - What, in a few lines, does each enhancement provide?
- **System Perspective**
  - Which functional areas of OS/2 Version 2 have been enhanced?
  - What enhancements have been added in each functional area?
  - Where, in this document, are the enhancements described in more detail?

This chapter provides an overview from both these perspectives on the OS/2 2.1 enhancements. Section 2.2, "System Perspective of OS/2 2.1 Enhancements" on page 10, also contains a list of the enhancements cross-referenced to the detailed descriptions.

A summary of the performance enhancements in OS/2 2.1 is also included in section 2.3, "Performance Improvements in OS/2 2.1" on page 14, at the end of the chapter. This includes graphs showing some areas of performance improvement of OS/2 2.1 over OS/2 2.0, such as 32-bit OS/2 applications and 16-bit Windows applications.

2.1 User Perspective of OS/2 2.1 Enhancements

2.1.1 Major Enhancements in OS/2 2.1

2.1.1.1 Improved Application Support

- *WIN-OS/2 3.1 Standard Mode support* - The WIN-OS/2 support is upgraded to provide support for Windows 3.1 applications (including Windows multimedia applications) running in Standard Mode, both seamless and full-screen. This replaces the WIN-OS/2 3.0 support.
- *WIN-OS/2 3.1 Enhanced Compatibility Mode support* - A new mode - Enhanced Compatibility Mode - has been added to the WIN-OS/2 support. This enables major Windows 3.1 enhanced-mode applications which do not require a specific type of device driver called a VxD, to run under WIN-OS/2 3.1. Mathematica for Windows and Omnipage Professional** are two such Windows enhanced-mode applications which run under WIN-OS/2 3.1.
- *Inclusion of MMPM/2 1.1* - Multimedia Presentation Manager/2 1.1 has now been included with OS/2 2.1. MMPM/2 provides 32-bit multimedia support code for OS/2 PM multimedia applications, including Software Motion Video.

2.1.1.2 Internal Operating System Upgrades

- *WIN-OS/2 3.1 Performance Enhancements* - Performance of Windows applications running under WIN-OS/2 has been substantially improved, especially when running in seamless mode.
- *32-bit PM Graphics Engine* - This rewrite of the graphics engine provides a sound basis for future powerful graphics. The immediate benefit of this is that it enables the new 32-bit seamless display drivers to be used.

2.1.1.3 Wider Hardware Support

- *Installation from CD-ROM* - OS/2 2.1 can now be installed from CD-ROM. This is a major improvement, and can save much time and energy. Two installation diskettes (both 3.5" and 5.25") are provided to start the CD-ROM installation process.
- *SCSI-attached CD-ROM support* - IBM and non-IBM SCSI-attached CD-ROM devices are now supported.
- *OS/2 2.1 Preloaded onto IBM and PC Manufacturer (PCM) PCs* - A range of IBM and non-IBM systems can now be supplied preloaded with OS/2 2.1.
- *Additional SCSI Adapter support* - IBM and non-IBM SCSI adapters are now supported.
- *Seamless 32-bit Display Drivers for XGA, XGA-2, SVGA (Tseng, Headland, Western Digital, Trident, ATI, Cirrus, and IBM VGA 256-color), 8514/A and VGA* - These provide seamless integration of OS/2 PM and Windows 3.1 applications on the OS/2 desktop.
- *A range of new and enhanced PM printer drivers, including the new high-speed printers* - are provided in OS/2 2.1.
- *Support for Windows 3.1 printer and display drivers* - The standard set of Windows 3.1 printer and display drivers have been included with OS/2 2.1, and these can be used with WIN-OS/2.
- *Advanced Power Management support* - OS/2 2.1 includes power-saving routines for notebooks and laptops which implement the APM specifications.
- *PCMCIA support* - OS/2 2.1 includes support routines for notebooks and laptops which use the PCMCIA bus for adapters.

2.1.2 Significant Enhancements in OS/2 2.1

2.1.2.1 Improved Application Support

- *Dual-Thread MVDM support* - This enables many DOS multimedia applications to run smoothly by providing threads inside the MVDM for simultaneously reading from disk and painting the screen or playing sound.
- *DOS_AUTOEXEC setting* - A specific DOS command file can be run automatically when a VDM is created.
- *Improved Clipboard and DDE support* - The Clipboard and DDE support in WIN-OS/2 has been improved, thus substantially increasing performance especially in exchanging data between WIN-OS/2 and OS/2 PM applications.
- *Ability to Start a DOS or OS/2 Application from a Windows Application* - It is now possible to "shell" to a DOS or OS/2 application from within a Windows application. This capability is especially useful for some Windows applications that depend on DOS utilities.

- *Inclusion of Windows 3.1 File Manager and most Accessories* - Most of the Windows 3.1 accessories, including the File Manager, Write and Paintbrush, have now been included with WIN-OS/2 3.1. See section 7.12, "Windows 3.1 Utilities and Accessories" on page 95 for a complete list of Windows accessories included.
- *Improved WIN-OS/2 Setup and Configuration* - The setup and configuration procedures for WIN-OS/2 applications have been substantially improved, especially for Clipboard and DDE support.

2.1.2.2 Internal Operating System Upgrades

- *Warning on ACL File Protections before Installation* - The OS/2 2.1 installation checks before it starts whether any files or directories are ACL protected, in order to ensure that installation can complete successfully,
- *Page Tuning Performance enhancements* - Improvements have been made to the memory working set in the Control Program, with consequent performance improvements in low-memory situations.
- *XCOPY enhancements* - The XCOPY utility has been enhanced to copy across system and hidden files, and to retain the read-only attribute.
- *Improved OLE support in WIN-OS/2 3.1* - The Object Linking and Embedding (OLE) support for compound documents has been improved as a result of upgrading WIN-OS/2 to use Windows 3.1 code.
- *Truetype** Fonts in WIN-OS/2 3.1* - The Truetype Font technology and core fonts have been included, also as a result of using the Windows 3.1 code.
- *ISO Font support* - This provides a set of ISO fonts in order to meet the new ISO 9241-3 ergonomic standard on appropriate hardware.
- *Improved INI file handling* - The modules responsible for storing and reading application and Workplace Shell data from the OS2.INI and OS2SYS.INI files have been improved and rewritten in 32-bit code, in order to be faster and more reliable.

2.1.2.3 Wider Hardware Support

- *Loadable ABIOS Support* - This feature provides the ability for ABIOS to be loaded into RAM from ABIOS files on disk, instead of using the ABIOS in ROM. This capability is necessary for the PS/2 Models 56 and 57, PS/2 Server 85, and ThinkPad* 700 series notebooks, and any other PS/2 systems which use loadable ABIOS rather than ABIOS in ROM.
- *Loadable ABIOS Installation* - This feature provides for the ABIOS files (which are shipped on the hardware reference diskette and on the system partition of the disk) to be installed on the boot disk as part of the OS/2 2.1 installation procedure. This capability is necessary for those PS/2 systems which use loadable ABIOS.
- *Enhancements to Selective Install program* - This feature has been enhanced for display driver, SCSI adapter, CD-ROM drive, printer installation, and also to allow the user to install WIN-OS/2 accessories as an option at installation time.
- *Display Driver Install program* - This program provides a simplified method for installation and configuration of XGA, XGA-2, SVGA and 8514/A display adapters, displays and resolution modes.

- *PS/2 Server 195 and 295 support* - OS/2 2.1 will run on the PS/2 Server 195 and 295 systems. Previously, Server 195 and 295 support was limited to OS/2 1.3.
- *Pentium** Exploitation* - OS/2 2.1 includes changes to exploit Intel's new Pentium chip. The first phase of this exploitation improves the performance of DOS sessions through the use of the Pentium virtual mode extensions.
- *Page Tuning Performance Enhancements* - The OS/2 2.1 developers have made some "under the cover" changes to improve the general performance of OS/2 2.1, especially in low-memory situations.
- *XGA-2 DMQS Override* - This feature enables the capabilities of advanced non-IBM displays to be exploited when they are attached to an XGA-2 adapter, by providing the ability to override the automatic XGA-2 display sensing (DMQS) with manual settings.
- *Enhanced Printer Installation* - User installation of additional printer drivers has been simplified in OS/2 2.1.

2.1.3 Minor Enhancements in OS/2 2.1

2.1.3.1 Improved Application Support

- *OS2VER file* - Some applications check for a specific version number of OS/2 2.0 before they run. The OS2VER file enables these applications to be told that they are running on OS/2 2.0 even though they are running on OS/2 2.1.
- *DPMI 1.0 Subset support* - The DPMI support has been upgraded to a subset of DPMI 1.0, enabling more DOS DPMI-based applications to run.
- *PC Support/400 support* - The new extended DOS version of PC Support/400 (V2R3), available during the second half of 1993, will run in an OS/2 2.1 VDM. This is due to the provision of multiple DPMI Client support as part of the DPMI 1.0 Subset support.
- *Multimedia support for Audio in WIN-OS/2 3.1* - Windows 3.1 Multimedia support for Audio is included, again as a result of using the Windows 3.1 code.
- *Workplace Shell Visual enhancements* - Some minor improvements have been made to the appearance of the Workplace Shell, including a new notebook appearance and new icons for CD-ROM drives.
- *Settings Notebook Drag/Drop enhancements* - It is now possible to add program objects to a folder menu and to change the icon of any object using drag/drop on the desktop.
- *Auto-lockup on System Startup* - The Workplace Shell can now be configured to enter lockup status automatically on system startup, thus ensuring the user enters a password before using OS/2 2.1 and providing an additional level of security.
- *Print Spooler Enhancements* - The print spooler has been enhanced to enable the user to vary the priority.

2.1.3.2 Internal Operating System Upgrades

- *Palette Manager for XGA, XGA-2, SVGA, 8514/A* - Palette Manager support is provided for 256-color adapters, enabling the application programmer to provide specific color shades to applications.

2.1.3.3 Wider Hardware Support

- *Brazilian Keyboard Support* - The new national Brazilian keyboard is supported.
- *Trackpoint II Support* - The in-keyboard pointing device on the ThinkPad 700 series of notebook computers is supported.
- *Support for Enhanced 2.88MB Diskette Drive* - The new 2.88MB diskette drive (6451271) that provides support for software eject and software lock/unlock is supported.
- *Support for 3.5" Enhanced Rewritable Optical Drive* - The new 3.5" Enhanced Rewritable Optical drive (6451295), with support for P-ROM (partial read-only memory) optical disks, and with software lock/unlock and eject functions, is supported.
- *Format Utility Enhanced for P-ROM Optical Disks* - The OS/2 Format utility has been enhanced to support formatting P-ROM optical disks in the new 3.5" Enhanced Rewritable Optical Drive.
- *Large Cursor on VGA LCD Displays* - On VGA LCD displays on notebooks, a larger cursor can be displayed to make it easier to see.
- *MSCDEX support* - The VCDROM virtual device driver now supports all the CD audio functions of the Microsoft** MSCDEX DOS device driver, thus enabling more DOS and Windows multimedia applications to run in VDMs.

2.1.4 Defect Removal in OS/2 2.1

OS/2 2.1 also includes substantial code optimization and defect removal. The APAR fixes are listed in Appendix D, "APAR Fixes in OS/2 2.00.1, Service Pak XR06055, and OS/2 2.1" on page 341.

2.2 System Perspective of OS/2 2.1 Enhancements

Figure 3 is a diagram of the OS/2 2.1 high-level system structure.

This is a simplified view of the structure of OS/2 2.1, but it may help with understanding how the various OS/2 2.1 components interrelate.

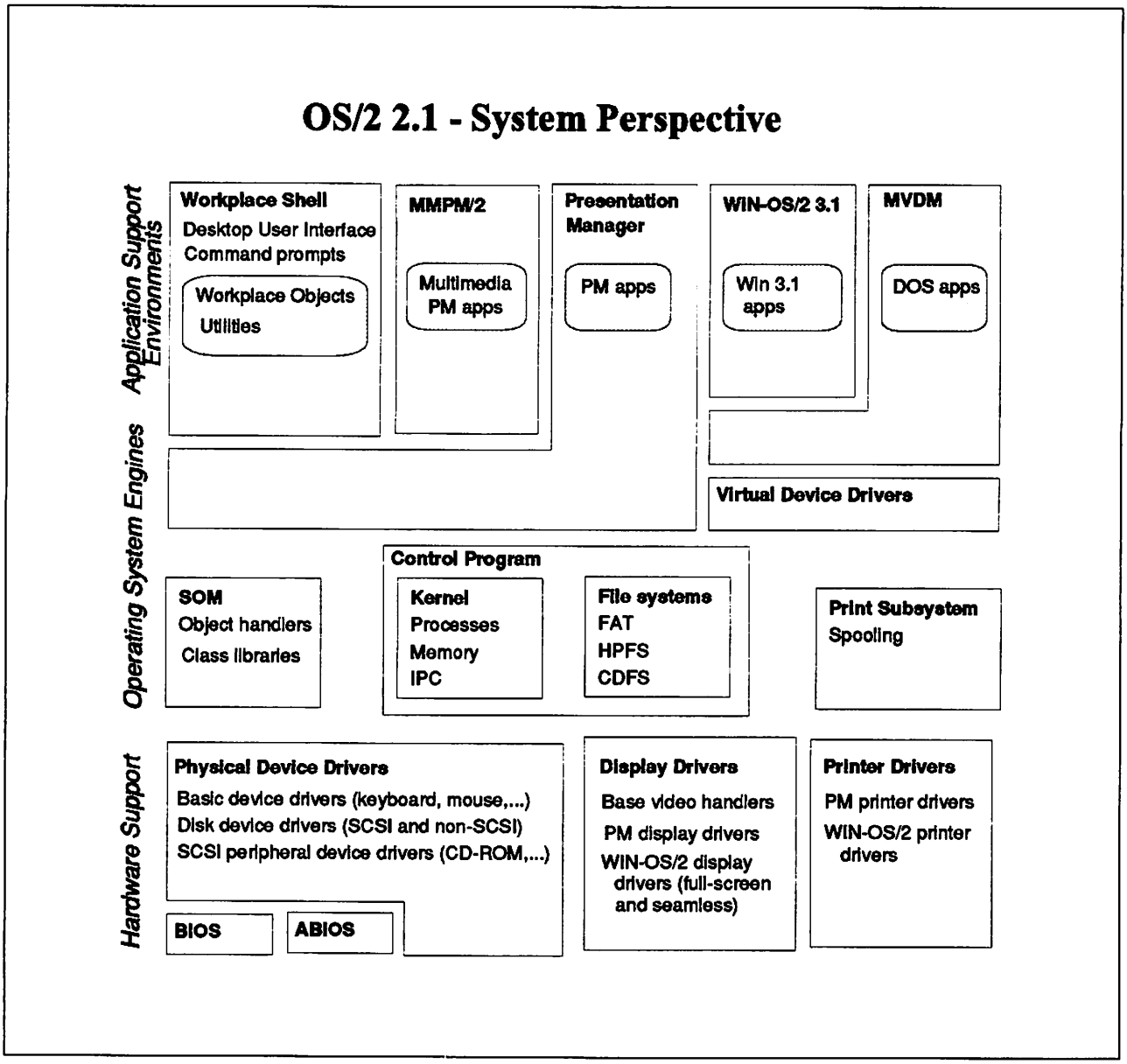


Figure 3. OS/2 2.1 - System Overview

In the previous section, the enhancements to OS/2 2.1 were divided into three groups:

- Application Support Environments
- Operating System Engines
- Hardware Support

These groups are shown as layers in this diagram.

The system perspective of OS/2 2.1 divides the structure into several functional components.

- BIOS (CBIOS)
- ABIOS
- Physical device drivers
- Display drivers
- Printer drivers
- Control Program
  - Kernel
  - File systems
- System Object Model
- Print subsystem
- Virtual device drivers
- MVDM (DOS support)
- WIN-OS/2
- Presentation Manager
- MMPM/2 (multimedia support)
- Workplace Shell
  - Desktop user interface
  - Command prompts
  - Utilities

2.2.1 OS/2 2.1 Enhancements - Summary Contents

The enhancements in OS/2 2.1 are now listed in tables divided according to the above subsystems and functional components, along with a reference to detailed discussions in this document.

Table 1. OS/2 2.1 Installation and Configuration Enhancements

Enhancement	Section
Installation from CD-ROM	3.1
Preloaded onto IBM and PCM systems	3.8
Enhancements to Selective Install program	3.2
Display Driver Install program	3.3
Warning on ACL File Protection before installation	3.5
Loadable ABIOS installation	3.4

Table 2. OS/2 2.1 Hardware Support Enhancements

Enhancement	Section
Additional SCSI adapter support	4.6
Additional CD-ROM support	4.7
Loadable ABIOS support	4.3
PS/2 Server 195 and 295 support	4.4.1
Brazilian keyboard support	4.4.2
Support for Enhanced 2.88MB diskette drive	4.4.3
Support for 3.5″ enhanced rewritable optical drive	4.4.4
Pentium exploitation	4.10

Table 3. OS/2 2.1 Control Program Enhancements	
Enhancement	**Section**
Page tuning performance enhancements	13.6
XCOPY enhancements	5.4
OS2VER file	5.2
Format Utility enhanced for P-ROM optical disks	5.3

Table 4. OS/2 2.1 MVDM Enhancements	
Enhancement	**Section**
Dual-thread MVDM support	6.1
DOS_AUTOEXEC setting	6.2
DPMI 1.0 Subset support	6.3
PC Support/400 (V2R3) support	6.3
MSCDEX support	6.4

Table 5. OS/2 2.1 WIN-OS/2 3.1 Support and Enhancements	
Enhancement	**Section**
WIN-OS/2 3.1 Standard Mode support	7.3
WIN-OS/2 3.1 Enhanced Compatibility Mode support	7.4
WIN-OS/2 3.1 performance enhancements	2.3.2, 13.5.4
Seamless WIN-OS/2 display drivers for XGA, SVGA (Tseng, Headland, Western Digital, Trident, ATI, Cirrus, and IBM Speedway), 8514 and VGA	7.9
Support for Windows 3.1 printer drivers	7.10
Improved Clipboard and DDE support	7.11
Ability to start a DOS or OS/2 application from a Windows application	7.8
Inclusion of Windows 3.1 File Manager and selected accessories	7.12
Improved WIN-OS/2 setup and configuration	7.13.1
Improved OLE support in WIN-OS/2 3.1	7.12
Truetype fonts in WIN-OS/2 3.1	7.12
Multimedia support for audio in WIN-OS/2 3.1	7.12

Table 6. OS/2 2.1 Presentation Manager Enhancements	
Enhancement	**Section**
32-bit PM Graphics Engine	2.3, 8.2
32-bit seamless PM display drivers for XGA, SVGA (Tseng, Headland, Western Digital, Trident, ATI, Cirrus, and IBM Speedway), 8514 and VGA	8.3, 4.9.1
ISO font support	8.5
Palette Manager for XGA, XGA-2, SVGA, 8514/A	8.4
XGA-2 DMQS override	3.7, 8.3.5

Table 7. OS/2 2.1 Workplace Shell Enhancements	
Enhancement	**Section**
Improved INI file handling	9.4
Workplace Shell visual enhancements	9.1
Settings notebook drag/drop enhancements	9.2
Auto-lockup on system startup	9.3

Table 8. OS/2 2.1 Print Subsystem Enhancements	
Enhancement	**Section**
Enhanced printer installation	10.1
Print Spooler enhancements	10.2
Additional and enhanced PM printer drivers	10.1

Table 9. OS/2 2.1 Enhanced Support for Laptops and Notebooks	
Enhancement	**Section**
Advanced Power Management support	11.1
PCMCIA support	11.2
Large cursor on VGA LCD displays	11.3
Trackpoint II support	11.4

Table 10. OS/2 2.1 Multimedia Support	
Enhancement	**Section**
Inclusion of MMPM/2 1.1 with OS/2 2.1	12.1.1
Software Motion Video	12.1.3
Multimedia device drivers	12.2.1
MMPM/2 applets	12.2.3
MMPM/2 utilities	12.2.5
MMPM/2 installation	12.3
Media Control Interface subsystem	12.4.1
Stream Programming Interface subsystem	12.4.2
Multimedia I/O Services subsystem	12.4.3
Additional multimedia controls	12.4.4
Applications for MMPM/2	12.6

2.3 Performance Improvements in OS/2 2.1

OS/2 2.1 includes performance improvements over OS/2 2.0. This is mainly in the areas of 32-bit OS/2 PM graphical applications and 16-bit Windows applications.

32-bit OS/2 PM graphical applications can take advantage of the 32-bit graphics engine and the new 32-bit display drivers, and can avoid having to pass through thunk layers in order to reach these components. See section 13.5.1, "32-Bit Graphics Engine" on page 197 for more details.

Windows applications can take advantage of the new WIN-OS/2 3.1 support, which combines the advantages of the Windows 3.1 code with the new seamless WIN-OS/2 display drivers. See section 13.5.4, "WIN-OS/2 3.1" on page 202 for more details.

Performance Disclaimer

The performance charts in this section have been compiled from performance tests run by the Personal Systems Programming development laboratory at Boca Raton in Florida. These tests have been run in a controlled environment, and therefore the results obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

In addition, performance data varies between applications and between different hardware configurations. The performance results presented here are for some commonly used applications running on some typical hardware configurations.

2.3.1 OS/2 2.1 Applications

Figure 4 on page 15 shows the relative performance of a common 32-bit OS/2 PM application (Lotus 1-2-3/G), running on both OS/2 2.0 and OS/2 2.1.

These tests were run on a mixture of PS/2 systems, with a range of display adapters. The performance index for OS/2 2.0 has been normalized to 1.00 for each display adapter; the graph thus shows the relative performance gains of OS/2 2.1 over OS/2 2.0 for each display adapter.

The performance of this 32-bit OS/2 application is between 18% and 24% faster under OS/2 2.1 than under OS/2 2.0.

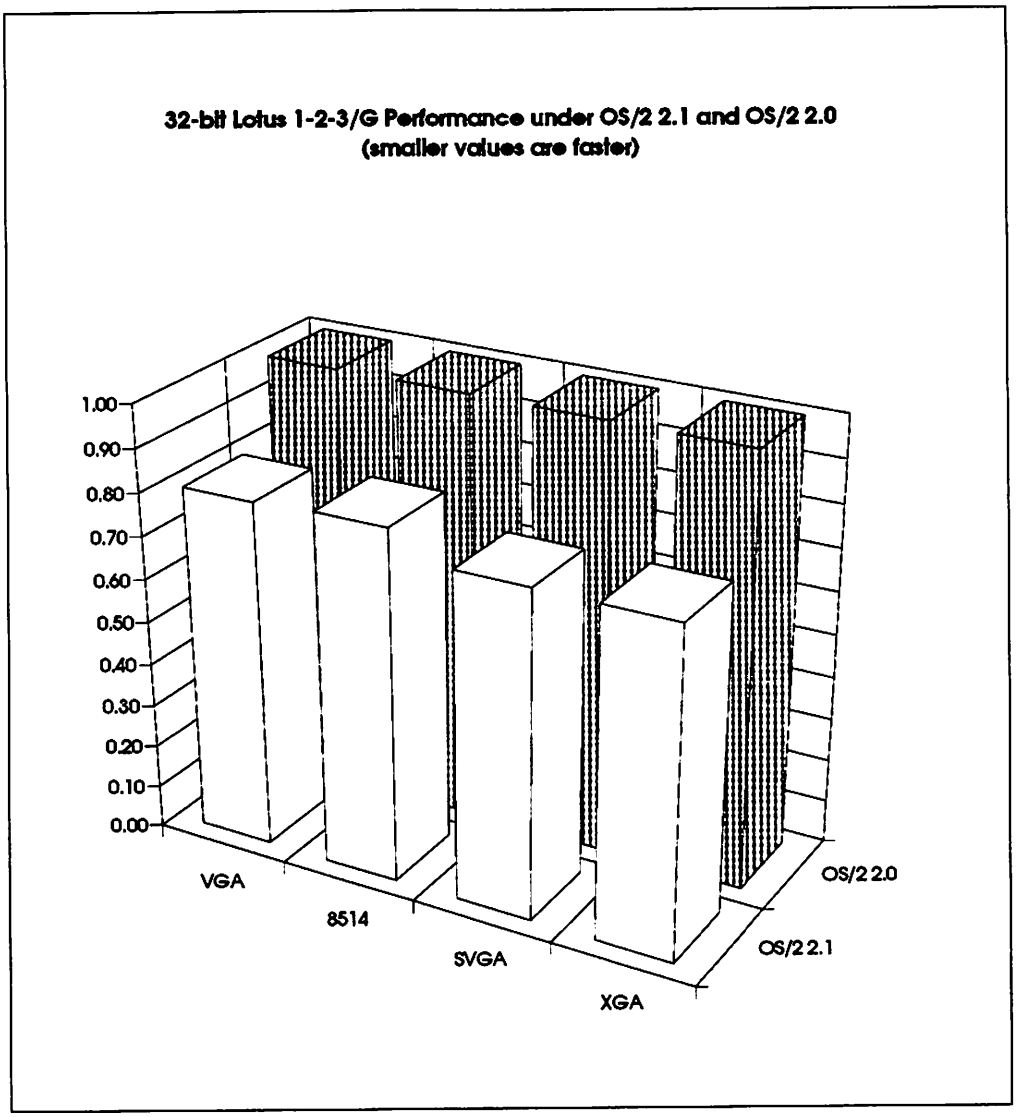


Figure 4. OS/2 2.1 Performance Improvements - 32-bit Lotus 1-2-3/G

2.3.2 WIN-OS/2 3.1 Applications

Figure 5 on page 16 shows the relative performance of a set of common 16-bit Windows applications (Microsoft Word 2.0a, Lotus Amipro 2.0, Microsoft Excel 4.0, Autocad 2.0, Describe 3.0), running under OS/2 2.1, OS/2 2.0 and Windows 3.1. Performance has been averaged over the applications in the set, and over various functions within the applications.

For OS/2 2.1 and OS/2 2.0, both full-screen and seamless modes are shown.

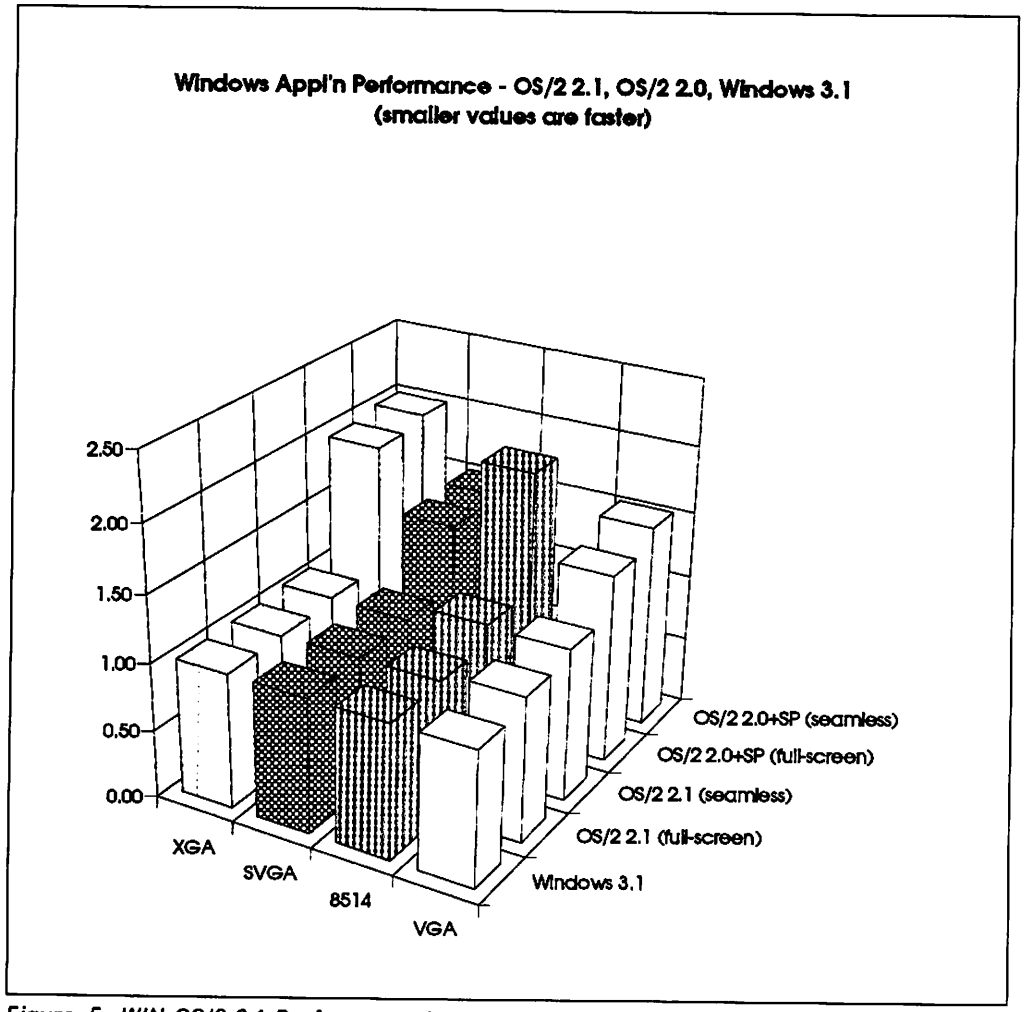


Figure 5. WIN-OS/2 3.1 Performance Improvements - Windows Applications

There is no value on these graphs for 8514 seamless on OS/2 2.0 with Service Pak XR06055 applied, since the 32-bit seamless 8514 driver was not released at that time.

These tests were run on a mixture of PS/2 systems, with 8MB memory and with a range of display adapters. The performance index for Windows 3.1 has been normalized to 1.00 for each display adapter; the graph thus shows the relative performance of OS/2 2.1 and OS/2 2.0 compared to Windows 3.1, for each display adapter.

Note that the performance of Windows applications under OS/2 2.1 is much better than under OS/2 2.0. The performance improvement is especially good in seamless mode.

Note also that the performance of Windows applications under OS/2 2.1 is now nearly the same as under Windows 3.1. The WIN-OS/2 3.1 support in OS/2 2.1 has been much improved over OS/2 2.0, and the performance is within a few percent in most cases, thus making it comparable with Windows 3.1.

Chapter 3. Installation and Configuration of OS/2 2.1

OS/2 2.1 is a licensed program product. The shrinkwrap package includes the *OS/2 2.1 Installation Guide*, which provides good documentation for installing OS/2 2.1, and should be the initial reference.

Although installation of OS/2 2.1 from diskette is fundamentally similar to installation of OS/2 2.0, there are enhancements and special features, and a new CD-ROM alternative has been provided.

This chapter focuses on these areas of difference, and on any special considerations for OS/2 2.1 installation.

Differences in OS/2 2.1 installation from OS/2 2.0 include:

- Installation of OS/2 2.1 from CD-ROM
- Enhancements to the Selective Install program, for:
  - SCSI adapter support
  - CD-ROM device support
  - Display driver support
  - Default printer installation
  - Control over WIN-OS/2 3.1 installation
- A new Display Driver Install program (DSPINSTL)

Special considerations for OS/2 2.1 installation include:

- Installation of OS/2 2.1 on loadable ABIOS systems
- Upgrade Installation of OS/2 2.1 on ACL-protected systems
- SVGA display driver installation and configuration
- XGA display driver configuration

Remote and CID-enabled installation of OS/2 2.1 have also been enhanced, and are described in the companion ITSC bulletins, especially *OS/2 Version 2.1 Remote Installation and Maintenance, GG24-3780-01* and *Automated Installation for CID Enabled OS/2 V2.1, GG24-3783-01*.

3.1 CD-ROM Installation of OS/2 2.1

Note

OS/2 2.1 introduces the ability to install from CD-ROM. This process is a much easier and faster way of installing OS/2 2.1 than from diskettes, and is suggested whenever a CD-ROM drive is available.

The CD-ROM installation process requires that the system be booted from two diskettes; these are supplied along with the CD-ROM and have some differences from the standard OS/2 2.1 boot diskettes. Booting from diskette is necessary in order to load the SCSI adapter and CD-ROM device drivers and the CD file system.

Installation from SCSI-based CD-ROM is supported by device drivers provided with OS/2 2.1; for a full list see section 4.7.2, "SCSI-Attached CD-ROM Support in OS/2 2.1" on page 52.

3.2 Enhancements to the Selective Install Program

The Selective Install program can be used automatically during initial OS/2 2.1 installation. It can also be started at any time subsequently in order to install additional OS/2 2.1 features, by double-clicking on the **Selective Install** icon in the **System Setup** folder (as shown in Figure 6). Selective Install can also be started by typing **Install** from the command prompt, as long as the boot drive is the default drive.

Before starting the Selective Install procedure, make sure that you do not have any Windows programs started, since OS/2 will not be able to complete the installation until all Windows programs have been closed.

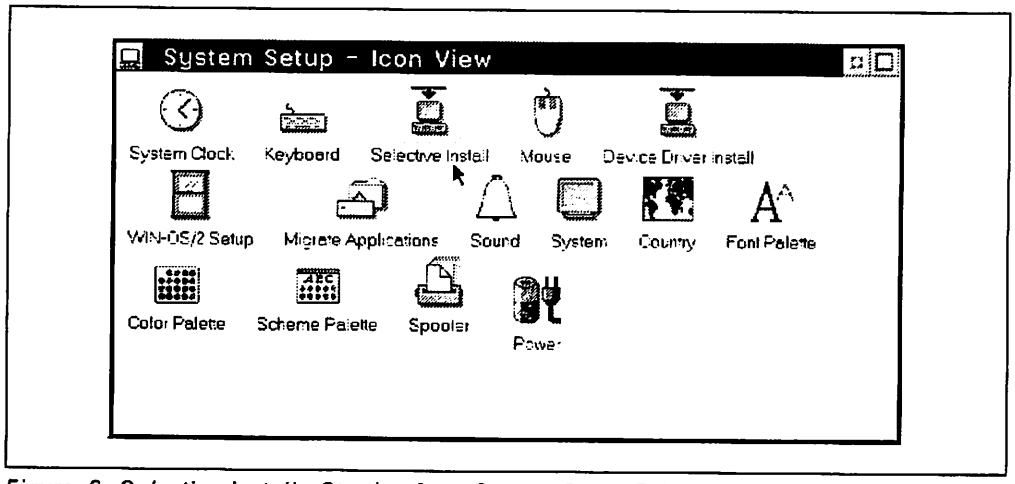


Figure 6. Selective Install - Starting from System Setup Folder

The Selective Install program has been significantly enhanced in OS/2 2.1 in the areas of:

- SCSI adapter support
- SCSI-based CD-ROM support
- Display driver support (for XGA, XGA-2, SVGA and 8514/A adapters)
- Default printer installation
- Control over WIN-OS/2 3.1 installation

When Selective Install is started (either automatically or explicitly), the System Configuration screen is displayed as in Figure 7 on page 19.

By selecting check-boxes on this screen and clicking on **OK**, further dialogs are displayed and these can be used to specify installation details for SCSI adapter support, SCSI-based CD-ROM device support, display drivers, and default printer support.

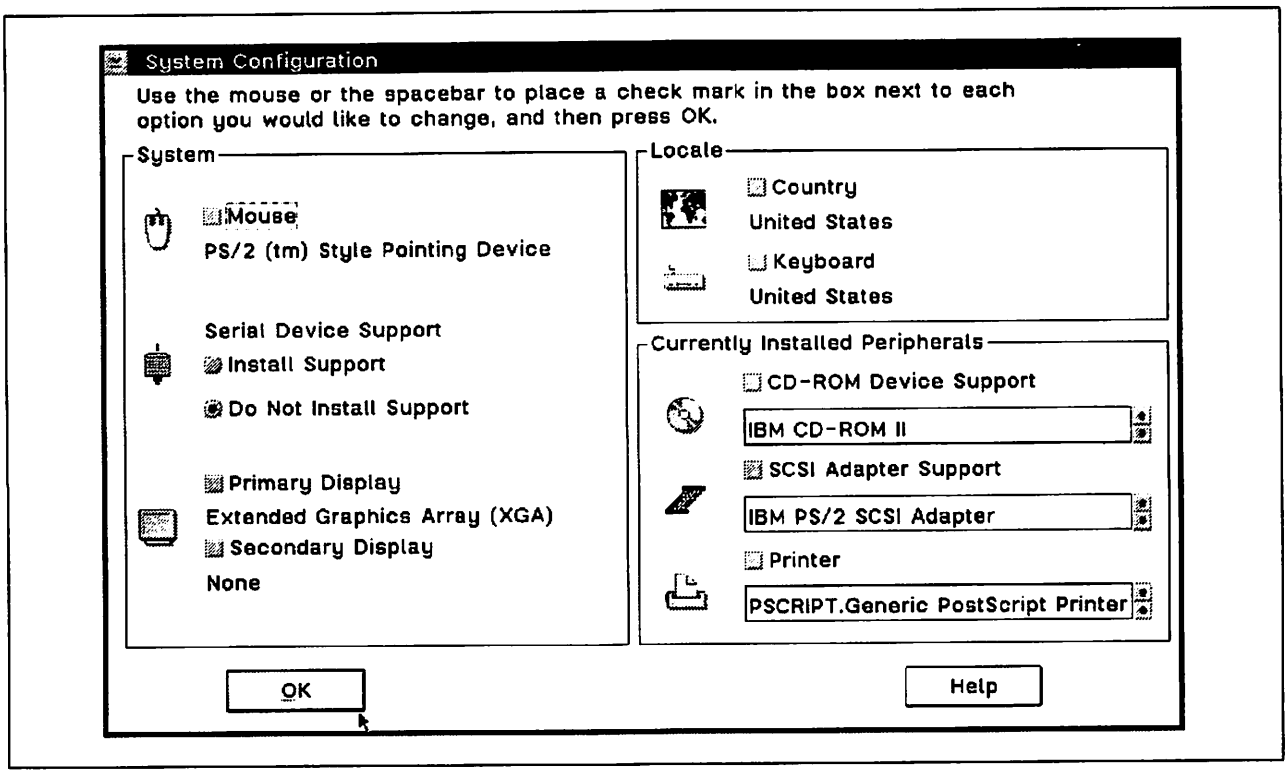


Figure 7. Selective Install - System Configuration

When all the options on this screen have been specified, click on **OK** in order to display the OS/2 Setup and Installation screen, as shown in Figure 8 on page 20.

This panel can be used to select optional features of OS/2 2.1 to install. Unlike the System Configuration screen, selecting a check box only selects the item for installation and does not automatically display a further dialog. Click on the **More...** push button in order to change the default parameters for installation of each option.

For example, the **More...** push button corresponding to WIN-OS/2 Support provides the way of controlling the selective installation of WIN-OS/2 3.1.

The disk space calculation in the bottom right corner of the screen, as shown in Figure 8 on page 20, shows how much free space is available on the disk, and also how much space is needed to install the features currently chosen for installation by the Selective Install program. (This calculation takes into account automatically the disk space required for installation of the essential parts of OS/2 2.1).

When all the options on this screen have been specified, click on **Install** in order to start the installation process. If this is the initial OS/2 2.1 installation, then the set of OS/2 2.1 installation diskettes will be prompted for. If this is a Selective Install, then only the required diskettes will be prompted for.

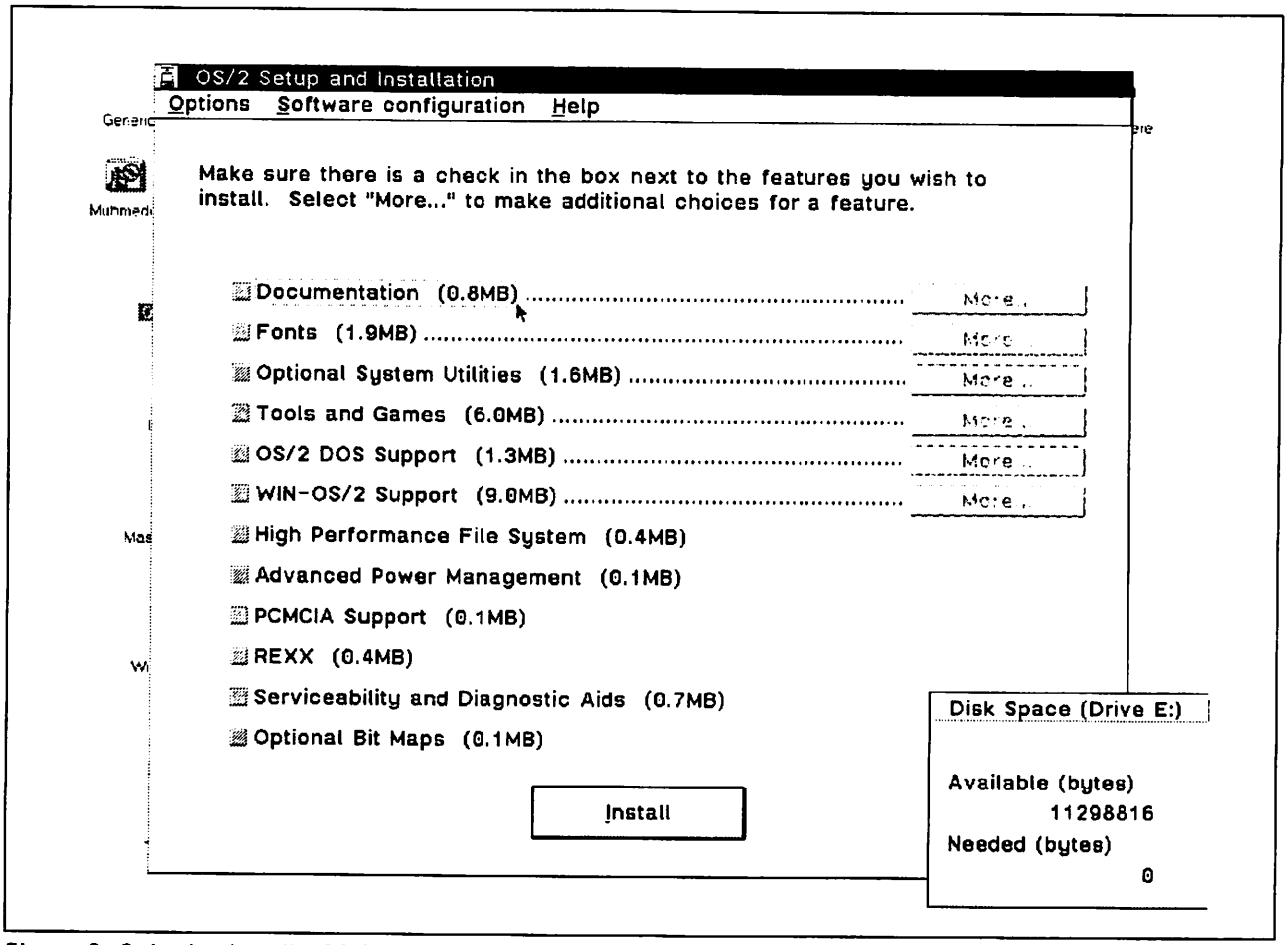


Figure 8. Selective Install - OS/2 Setup and Installation

At the end of the installation, the system will normally need to be rebooted for the changes to take effect (since new device drivers have to be loaded). If Selective Install was explicitly started, then a message box will be displayed as shown in Figure 9.

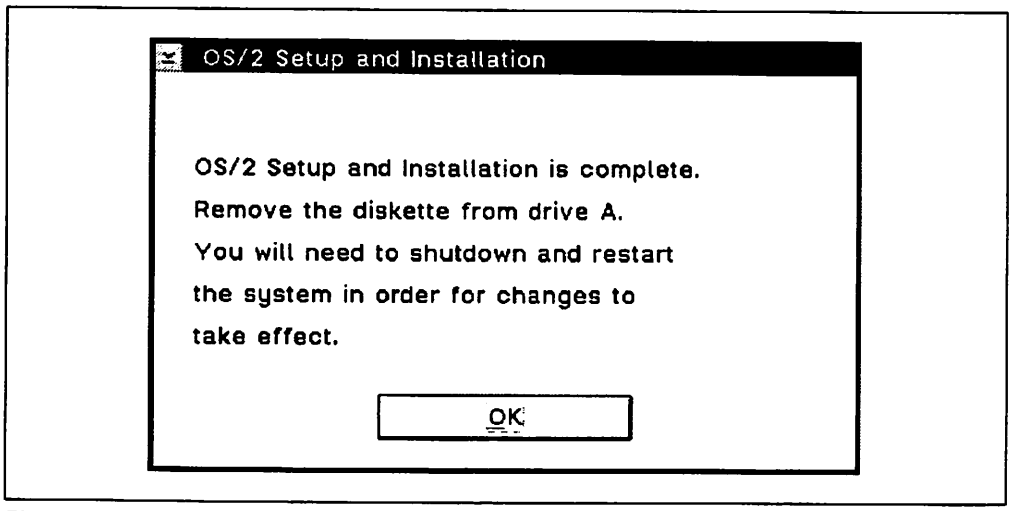


Figure 9. Selective Install - Reboot Message

3.2.1 CD-ROM Support Installation using Selective Install

CD-ROM drive support can be installed by following the instructions below, starting from the System Configuration screen (as shown in Figure 7 on page 19).

1. Select the check box beside the **CD-ROM Device Support** option.
2. Click on **OK** and the Select CD-ROM Device(s) window will be displayed, as in Figure 10.

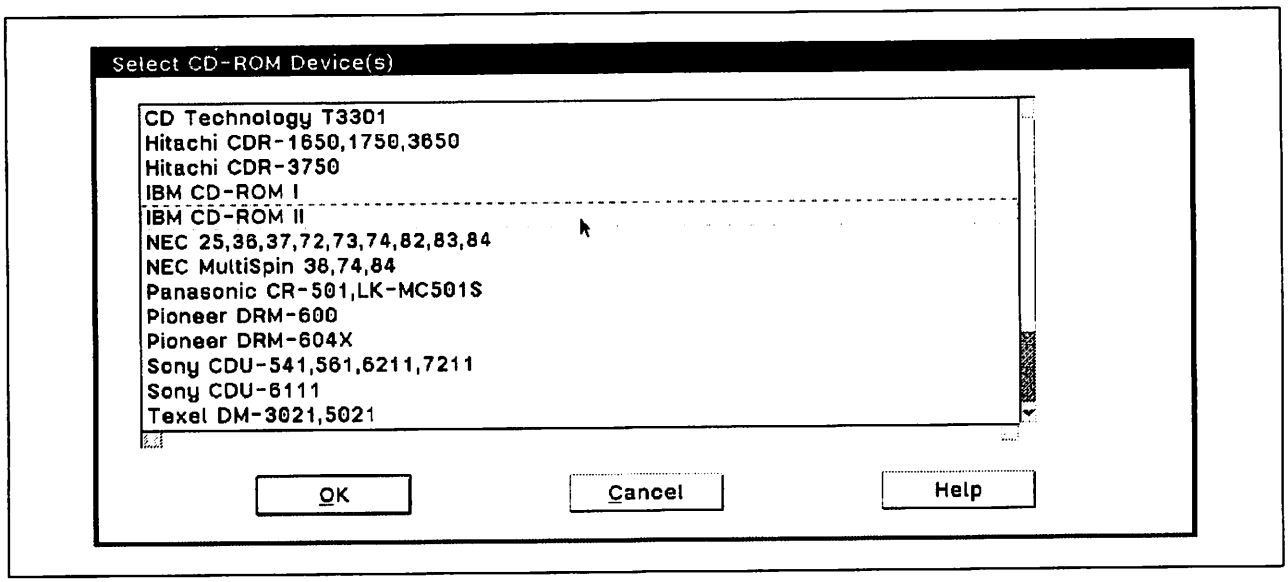


Figure 10. Selective Install: Select CD-ROM Device(s)

3. Select the list box item(s) for the CD-ROM devices which you wish to install support for.
4. Click on **OK** and the System Configuration window will be re-displayed.
5. When all the options on the System Configuration screen have been set, Selective Install moves on to the OS/2 Setup and Installation screen as shown in Figure 8 on page 20.
6. When all the options on the OS/2 Setup and Installation screen have been set, click on **Install** to start the installation process.

The SCSI-based CD-ROM support has been architected with a "pluggable" design, which allows additional SCSI-based CD-ROM drivers to be manually installed as and when they become available from third-parties following the general availability of OS/2 2.1.

Warning

There are some combinations of SCSI adapter cards and CD-ROM drives that are not supported. Please check the list in *OS/2 2.1 Using the Operating System* to check that your combination of adapter and CD-ROM drive is supported under OS/2 2.1.

3.2.2 SCSI Adapter Support Installation Using Selective Install

SCSI adapter support can be installed by following the instructions below, starting from the System Configuration screen (as shown in Figure 7 on page 19).

1. Select the check box beside the **SCSI Adapter Support** option.
2. Click on **OK** and the Select SCSI Adapters window will be displayed, as in Figure 11.

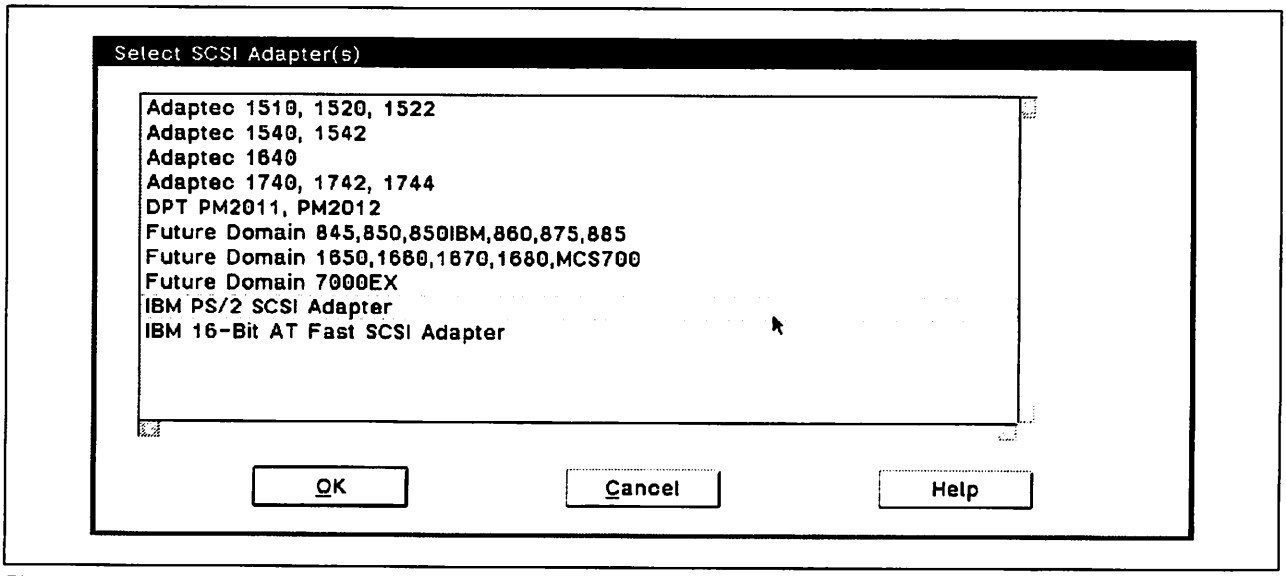


Figure 11. Selective Install: Select SCSI Adapters

3. Select the list box item(s) for the SCSI adapters which you wish to install support for.
4. Click on **OK** and the System Configuration window will be re-displayed.
5. When all the options on the System Configuration screen have been set, Selective Install moves on to the OS/2 Setup and Installation screen as shown in Figure 8 on page 20.
6. When all the options on the OS/2 Setup and Installation screen have been set, click on **Install** to start the installation process.

3.2.3 Display Driver Installation Using Selective Install

Note

If you have an SVGA adapter installed, then the initial installation of OS/2 2.1 only installs VGA support. This is because MVDM DOS support is not available during the initial installation of OS/2 2.1, and so it is not possible to run the SVGA DOS query program which establishes the capabilities of the SVGA adapter and display.

To install SVGA adapter support for resolutions higher than VGA (640x480x16), you should reinstall the SVGA display drivers after OS/2 2.1 installation has completed.

This can be done using either Selective Install or DSPINSTL. Selective Install should be used if ISO fonts are required; otherwise we recommend using DSPINSTL.

Display driver support can be installed by following the instructions below, starting from the System Configuration screen (as shown in Figure 7 on page 19).

1. Select the check box beside the **Primary Display** option.
2. Click on **OK** to display the **Primary Video Adapter** window as in Figure 12.

   This window will list the types of video connections that OS/2 can support.

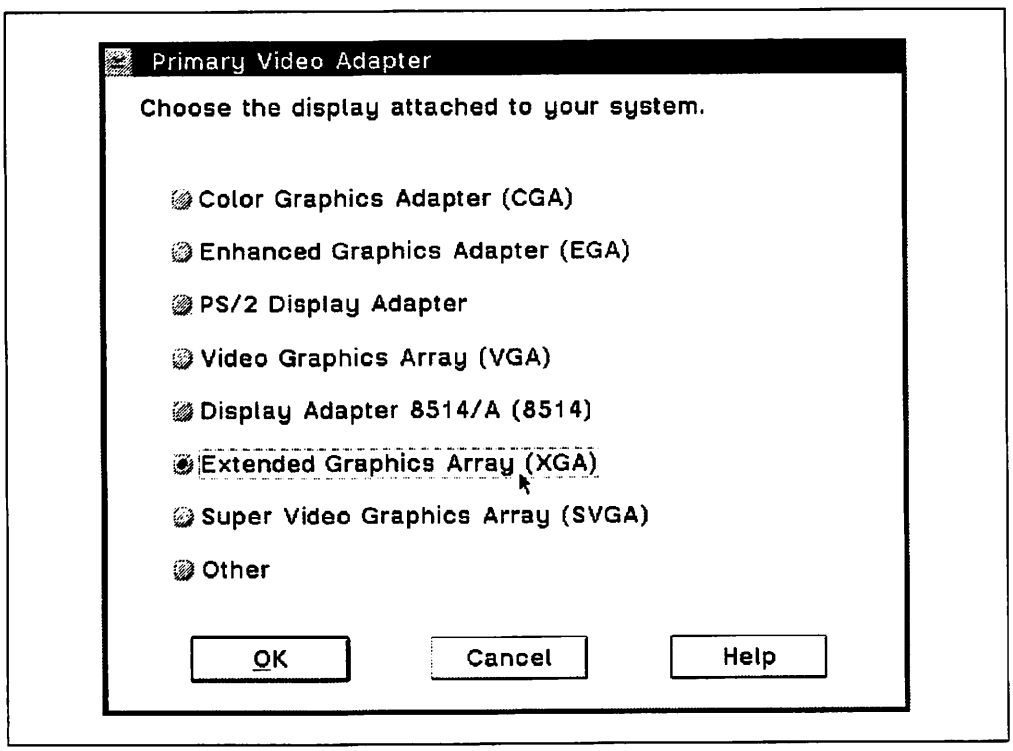


Figure 12. Selective Install: Primary Video Adapter

At this stage, OS/2 does not check the video adapter to determine if the adapter is supported by OS/2. This check will be done at the end of the installation, just before OS/2 2.1 prompts for the display driver diskettes. To

ensure your adapter is supported by OS/2, please check to make sure your adapter is in the list of SVGA adapters in Table 17 on page 54.

3. Select the radio button for the **SVGA** option or other display adapter installed on your system.
4. Click on **OK** and the **System Configuration** window will be re-displayed, as in Figure 13.
5. When all the options on the System Configuration screen have been set, Selective Install moves on to the OS/2 Setup and Installation screen as shown in Figure 8 on page 20.

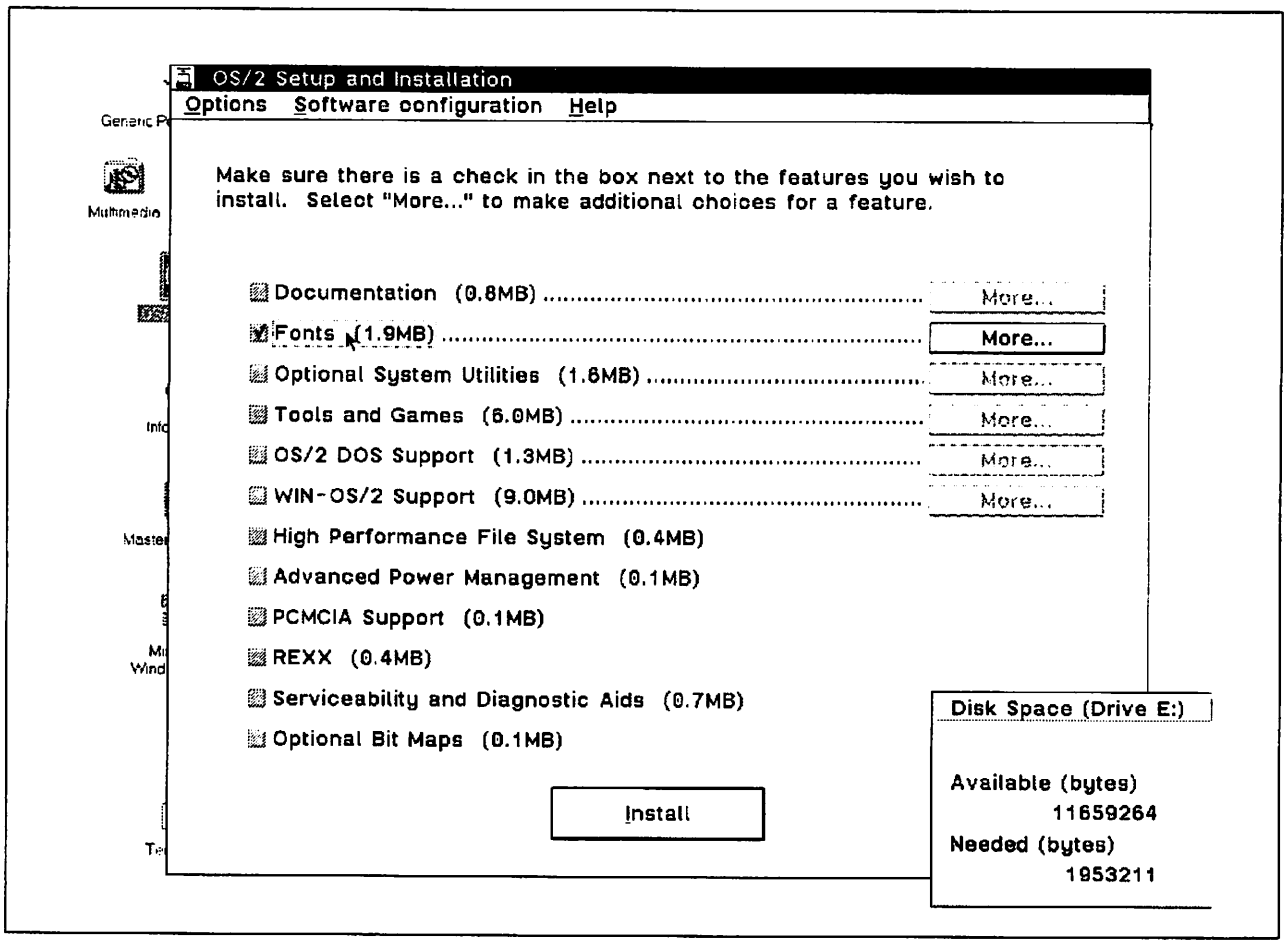


Figure 13. Selective Install - OS/2 Setup and Installation for Fonts

6. High-resolution fonts are automatically installed. If you want to install the ISO fonts, then click on **More...** opposite the Fonts check box, and ensure that all the fonts are selected in the Fonts window, as in Figure 14 on page 25.

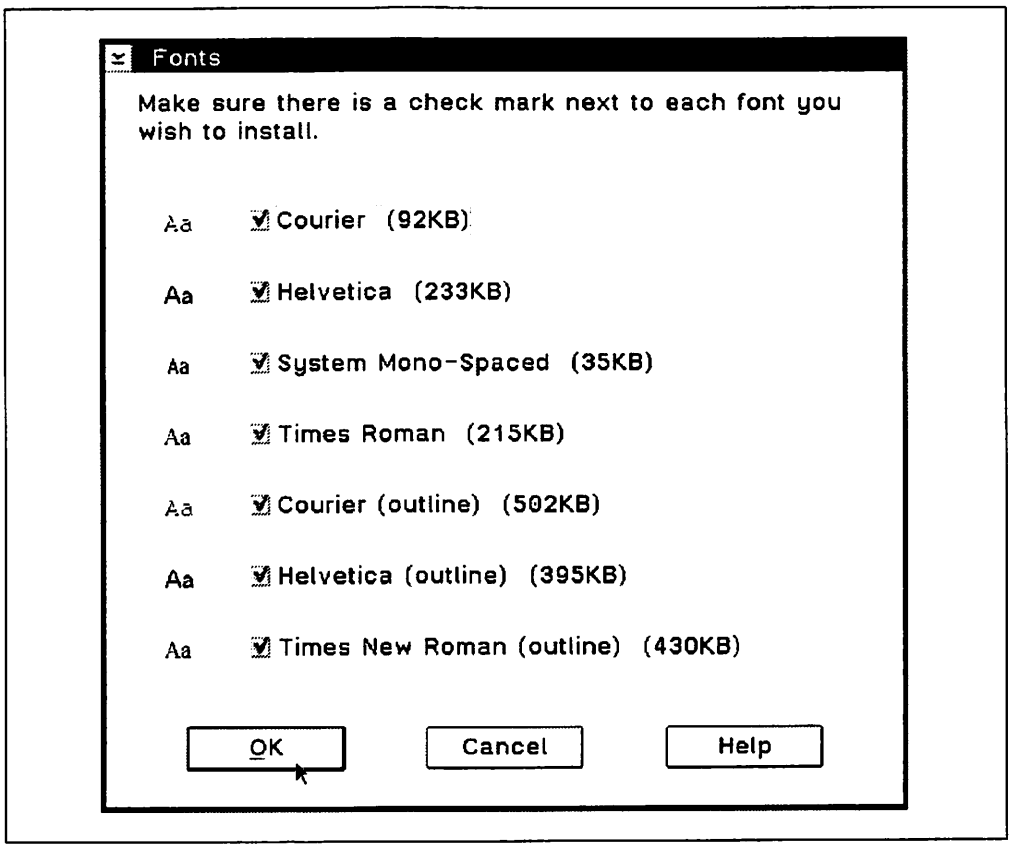


Figure 14. Selective Install - Fonts

7. Click on **OK** to return to the OS/2 Setup and Installation screen.
8. When all the options on the OS/2 Setup and Installation screen have been set, click on **Install** to start the installation process.
9. OS/2 will now check to establish which display adapter is installed in the system, and look at the chip set to determine which resolution it will support.

> **Note**
>
> For SVGA, there are different display drivers for each resolution.
>
> For XGA/XGA-2, one display driver supports all resolutions.

At this stage, if you have an SVGA adapter, the **Monitor Configuration** screen is displayed, as shown in Figure 15 on page 26. You will be asked whether you want to install a Display Adapter Utility program from diskette,

Some SVGA display adapters need to use a DOS Display Adapter utility program so that the adapter can know the refresh rate and possible resolutions of the monitor and configure it correctly. This utility program is normally provided with the SVGA display adapter and supplied on diskette.

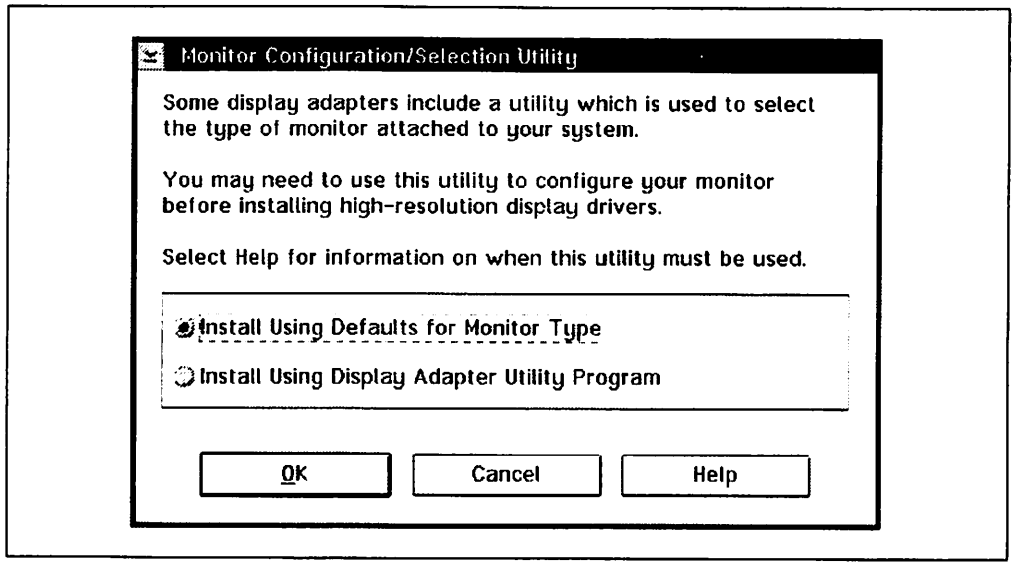


Figure 15. Selective Install - SVGA Monitor Configuration Screen

10. If your SVGA adapter requires this utility, then select **Install using Display Adapter Utility program**; otherwise choose **Install Using Defaults for Monitor Type**.
11. If you have chosen to install the Display Adapter utility program, you will be then be prompted to insert a diskette or specify a different path, as in Figure 16.

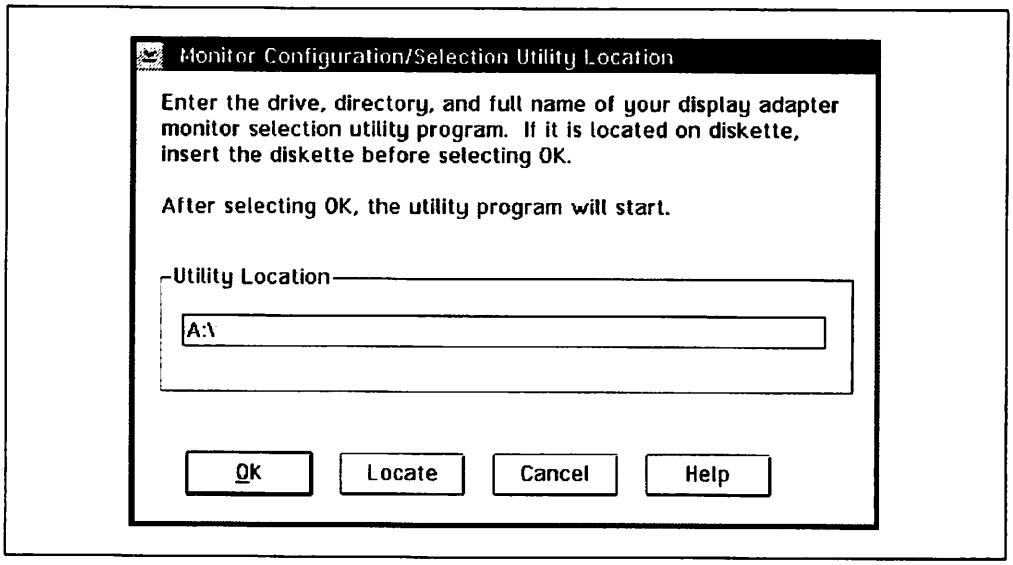


Figure 16. Selective Install - Diskette Prompting for DOS Monitor Utility

12. Specify the path as appropriate and insert the diskette if necessary, and then click on **OK**.
13. Selective Install of the display driver then continues by prompting you to choose the display resolution, as in Figure 17 on page 27.

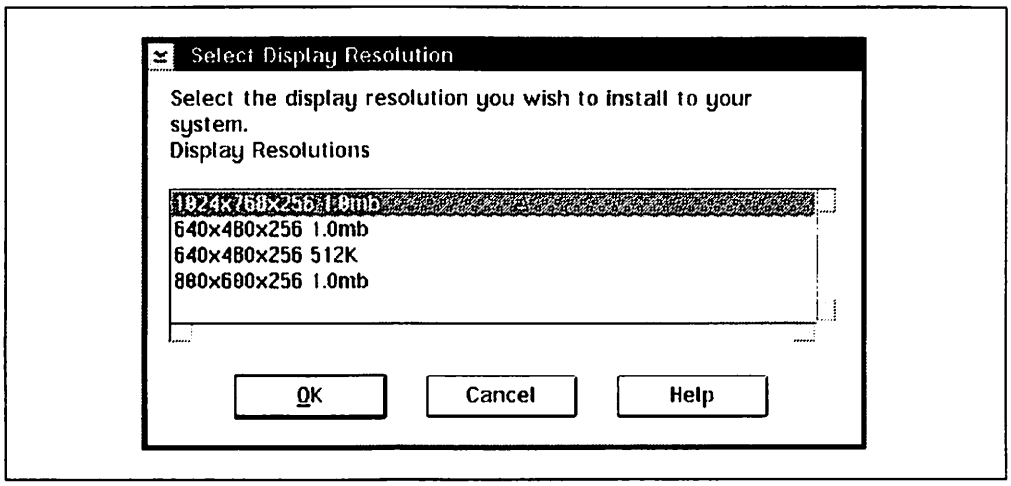


Figure 17. Selective Install - Select Display Resolutions

14. Select the list box item for the display resolution you require, and click on **OK** to install the appropriate display drivers.

Note

The Display Driver Install program, DSPINSTL, can also be used to install and configure display drivers.

If only display drivers are being installed, then the Display Driver Install program is easier and faster to use than Selective Install, and should normally be used. See section 3.3, "Display Driver Install Program" on page 30 for more details about DSPINSTL.

However, Selective Install should be used if ISO Fonts are required, or if CGA or EGA adapter support is being installed, since DSPINSTL does not support installation of these options.

3.2.4 Printer Installation Using Selective Install

Printer Installation is discussed in more detail in Chapter 10, "Print Subsystem Enhancements" on page 129.

The default printer driver can be installed by following the instructions below, starting at the System Configuration screen (as shown in Figure 7 on page 19).

1. Select the check box beside the **Printer** option.
2. Click on **OK** and the Select Printer(s) window will be displayed, as in Figure 18 on page 28.

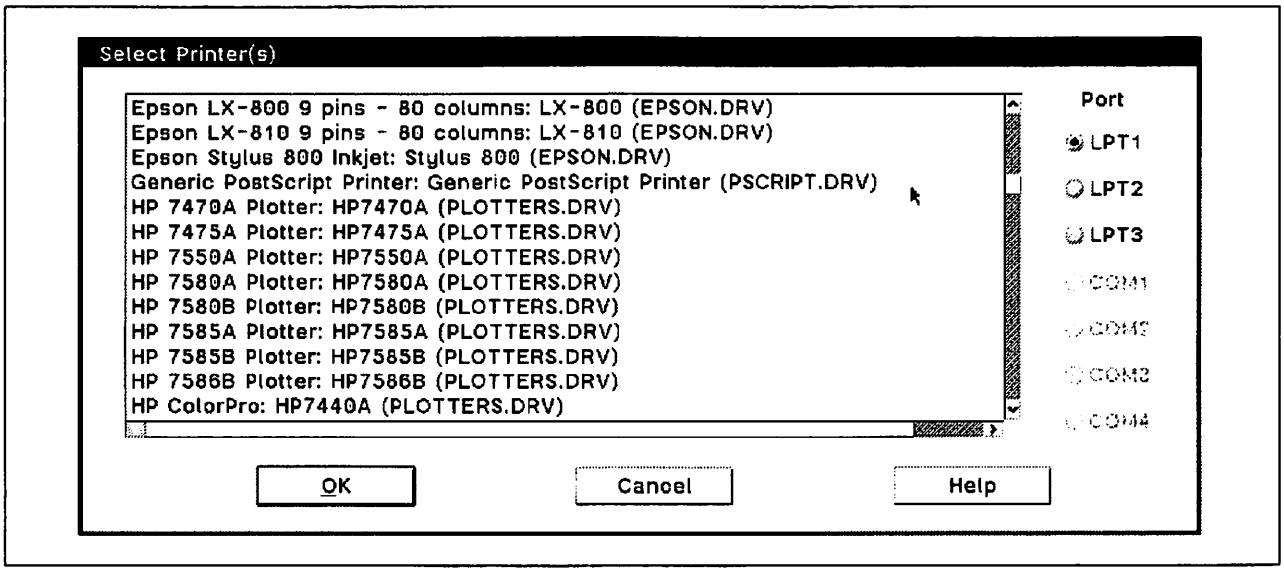


Figure 18. Selective Install - Select Printer(s)

3. Click on the printer you wish to have installed as your default printer.
4. Click on **OK** and the System Configuration window will be re-displayed.
5. When all the options on the System Configuration screen have been set, Selective Install moves on to the OS/2 Setup and Installation screen as shown in Figure 8 on page 20.
6. When all the options on the OS/2 Setup and Installation screen have been set, click on **Install** to start the installation process.

3.2.5 Selective WIN-OS/2 3.1 Installation

Two changes have been made in order to increase the flexibility of WIN-OS/2 3.1 installation, and to avoid running out of space when installing OS/2 2.1 on small disk partitions.

- It is now possible to install WIN-OS/2 3.1 onto a different drive from the rest of OS/2 2.1. This can be useful in cases where the system has two or more smaller disk partitions, since the disk requirements of OS/2 2.1 have increased over OS/2 2.0.
- The Windows applets can be selectively installed. The default is to install all the applets. If the Windows applets are not selected for installation, then this will reduce the amount of disk space needed for WIN-OS/2 3.1.

On the OS/2 Setup and Installation screen, ensure that the check box for **WIN-OS/2 Support** is selected, as shown in Figure 19 on page 29. DOS support is automatically installed if WIN-OS/2 support is selected.

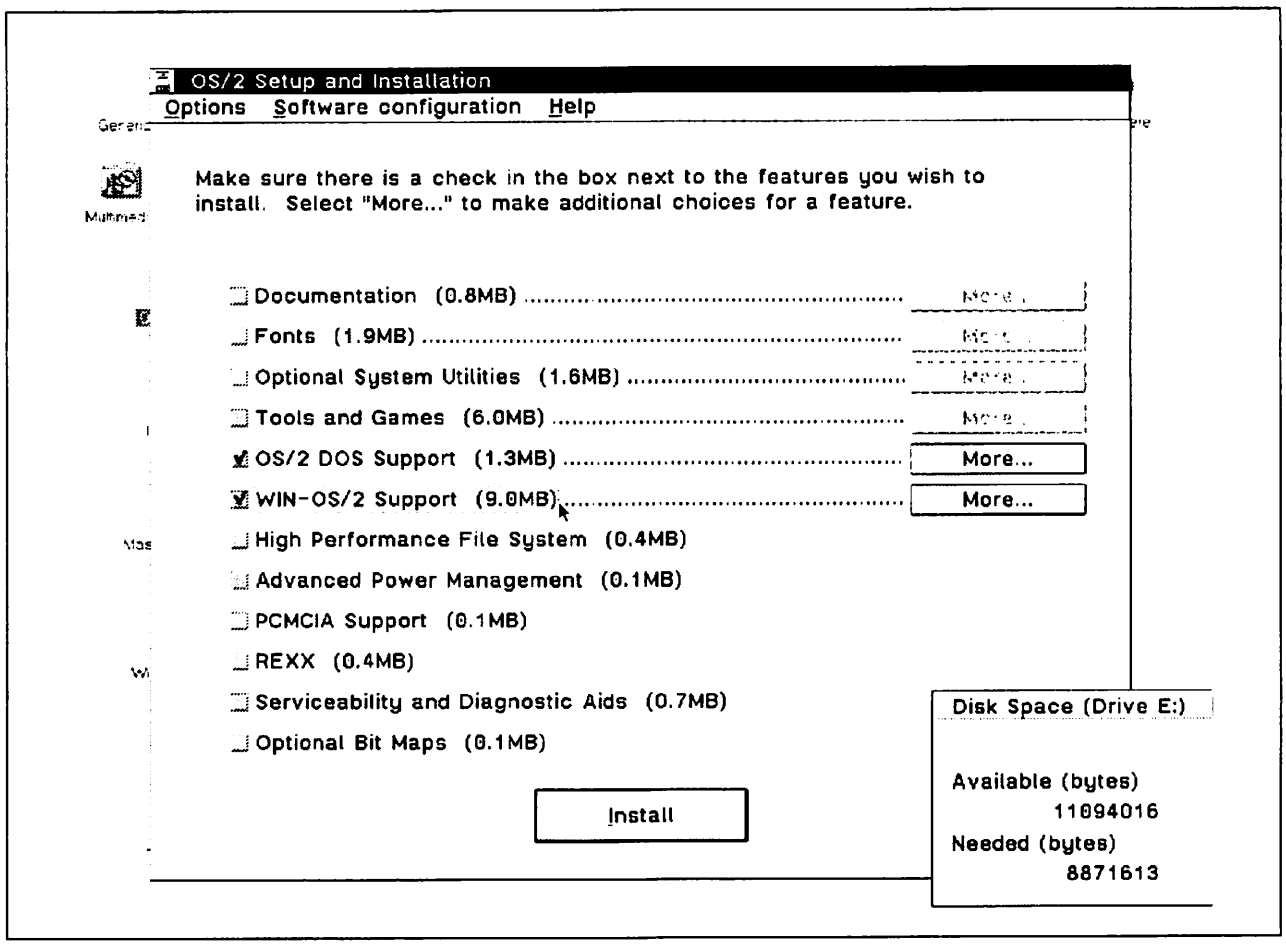


Figure 19. Selective Install - Choosing DOS and WIN-OS/2 Installation

Then click on **More...** in order to display the WIN-OS/2 support screen as shown in Figure 20.

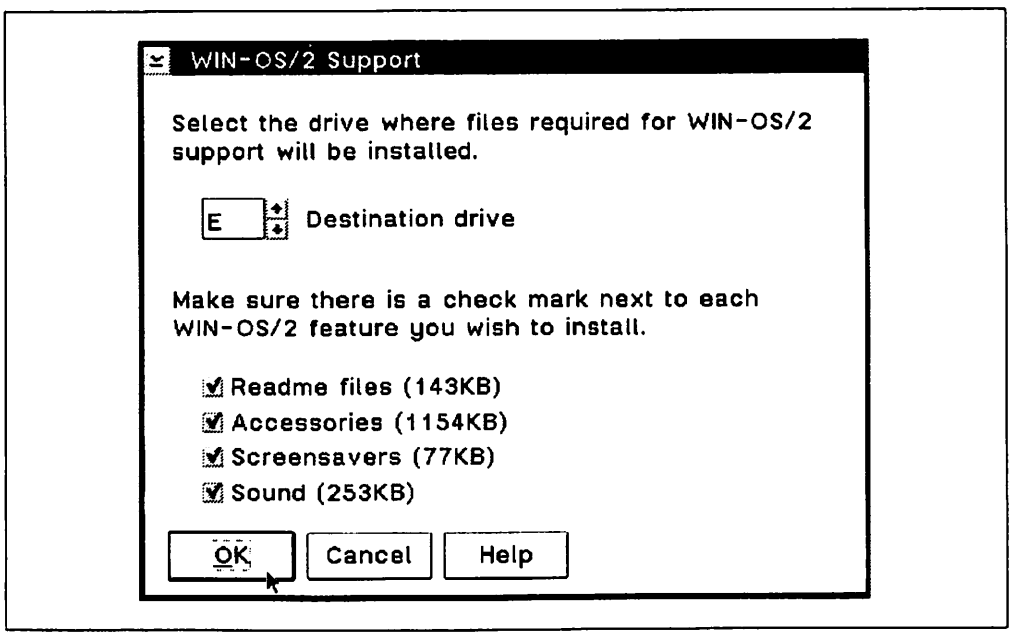


Figure 20. Selective Install - WIN-OS/2 3.1 Installation parameters

By default, all the WIN-OS/2 3.1 files will be installed, to the same disk drive as the rest of OS/2 2.1.

In order to change the drive on which WIN-OS/2 3.1 will be installed, perhaps because there is insufficient room on this drive, use the **Destination drive** spin button to choose another drive.

In order to avoid installing the optional WIN-OS/2 3.1 files, deselect the check box against any or all of the optional WIN-OS/2 features.

3.2.6 Selective APM and PCMCIA Installation

The OS/2 Setup and Installation screen also contains options for installing Advanced Power Management (APM) support, and PCMCIA adapter support.

Advanced Power Management provides support in the operating system to extend the battery life of laptop and notebook computers. The APM support check box will be automatically selected if your system has APM capability. For more details about Advanced Power Management, see section 11.1, "Advanced Power Management (APM) Support" on page 147.

PCMCIA is a standard for credit-card size adapters, typically used in laptop and notebook computers. You should check that PCMCIA support is selected if your system supports PCMCIA, as it is not automatically selected. PCMCIA will, however, be installed if a full install is selected. For more details about PCMCIA, see section 11.2, "PCMCIA Support" on page 153.

3.3 Display Driver Install Program

The Display Driver Install program (DSPINSTL) provides an alternative method of installing display drivers, which is easier and faster than Selective Install if only display drivers are being installed.

Note

Selective Install should be used instead of DSPINSTL if ISO fonts are required, or for CGA or EGA display driver installation.

The Display Driver Install program can be started by typing **DSPINSTL** at an OS/2 command line.

Before starting this procedure make sure that you do not have any Windows programs started, since OS/2 will not be able to complete the installation until all Windows programs have been closed.

To install and configure display drivers, follow these instructions:

1. On any OS/2 command prompt window type in the command **DSPINSTL** and press **Enter**. The Display Driver Install window is displayed, as in Figure 21 on page 31.

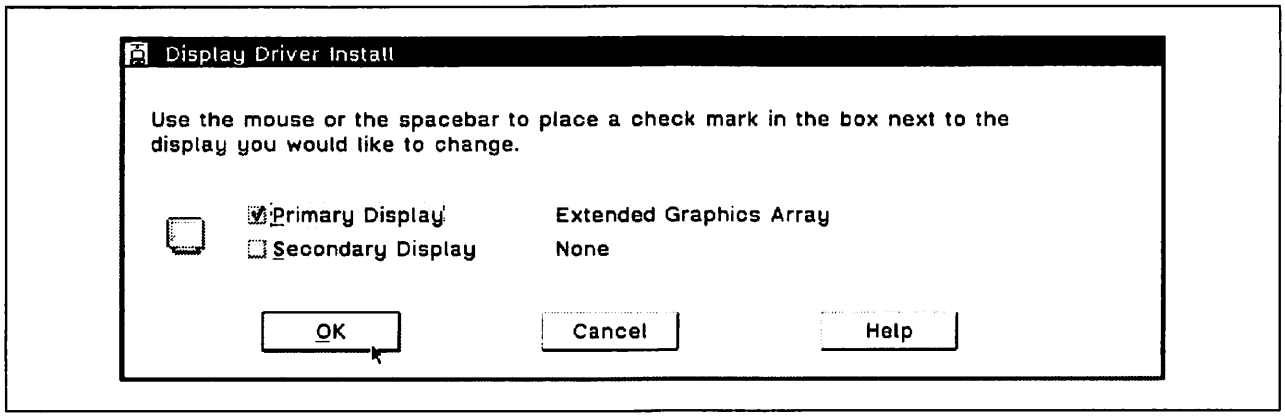


Figure 21. DSPINSTL - Display Driver Install

2. Select the check box beside the **Primary Display** option.

   If you are planning to connect two displays make sure you configure the primary display for the higher resolution.

3. Click on **OK** and the Primary Display Adapter Type window will be displayed, as in Figure 22

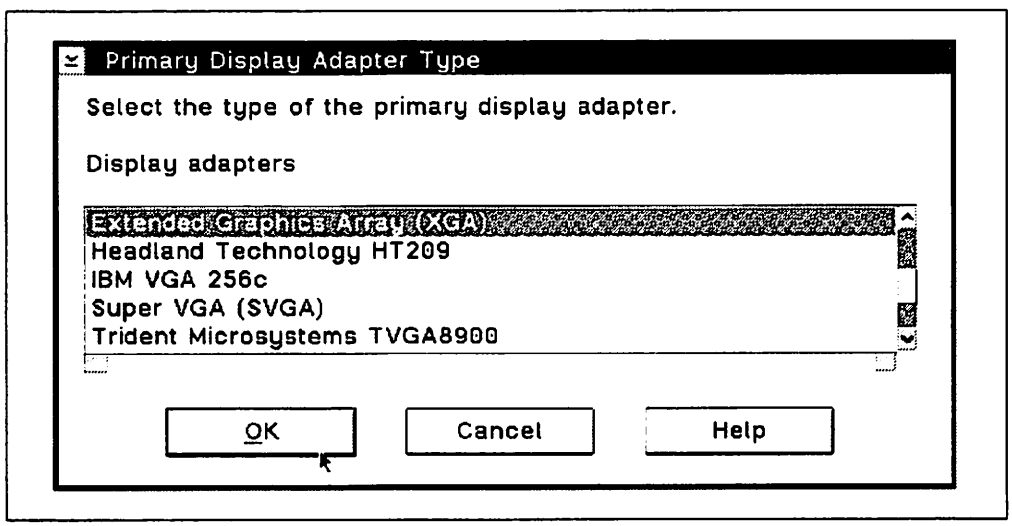


Figure 22. DSPINSTL - Primary Display Adapter Type

   OS/2 will now check your system to see which adapter is installed and will preselect the driver for you.

4. Accept the preselected display driver by clicking on **OK**, or select another display driver.

5. OS/2 will now check your video adapter to determine what resolutions can be supported.

   At this stage, if you have an SVGA adapter, the **Monitor Configuration** screen is displayed, as shown in Figure 23 on page 32. You will be asked whether you want to install a Display Adapter Utility program from diskette,

   Some SVGA display adapters need to use a DOS Display Adapter utility program so that the adapter can know the refresh rate and possible resolutions of the monitor and configure it correctly. This utility program is normally provided with the SVGA display adapter and supplied on diskette.

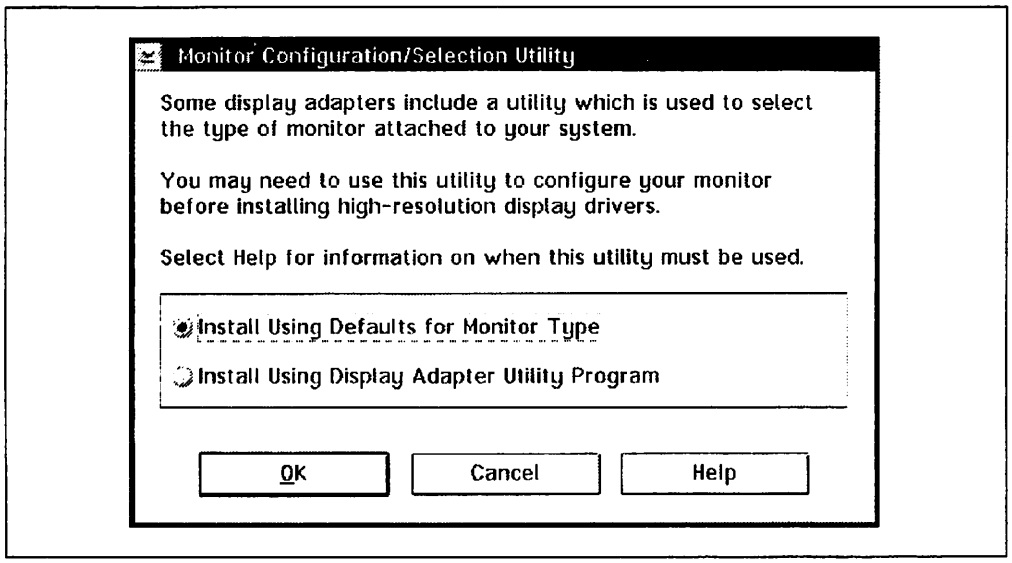


Figure 23. DSPINSTL - SVGA Monitor Configuration Screen

6. If your SVGA adapter requires this utility, then select **Install using Display Adapter Utility program**, otherwise choose **Install Using Defaults for Monitor Type**.
7. If you have chosen to install the Display Adapter utility program, you will be then be prompted to insert a diskette or specify a different path, as in Figure 24.

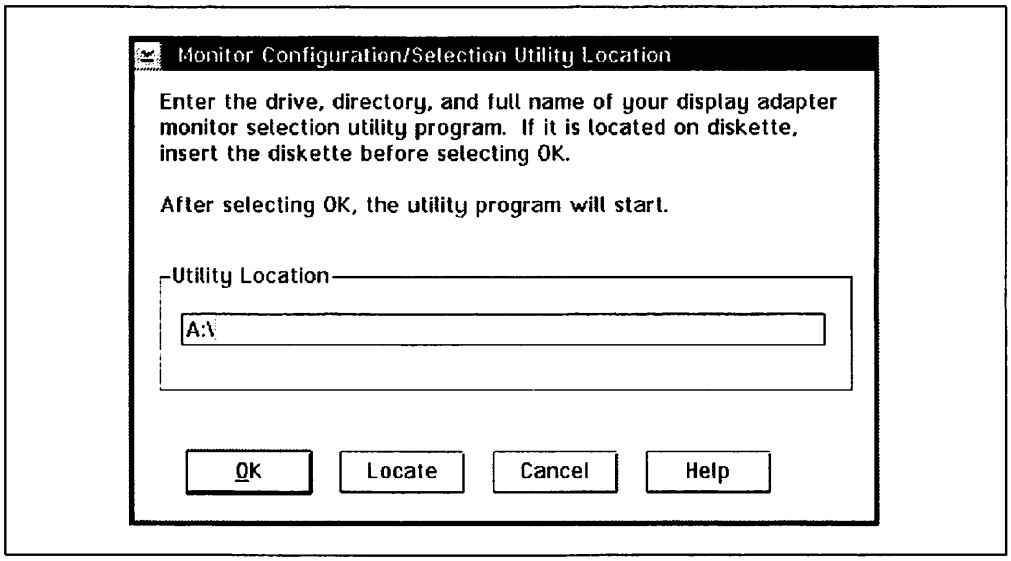


Figure 24. DSPINSTL - Diskette Prompting for DOS Monitor Utility

8. Specify the path as appropriate and insert the diskette if necessary, and then click on **OK**.
9. The Display Driver Install program then queries the adapter and monitor to find out the possible resolutions. You will then be presented with a screen prompting you to choose the display resolution, as in Figure 25 on page 33.

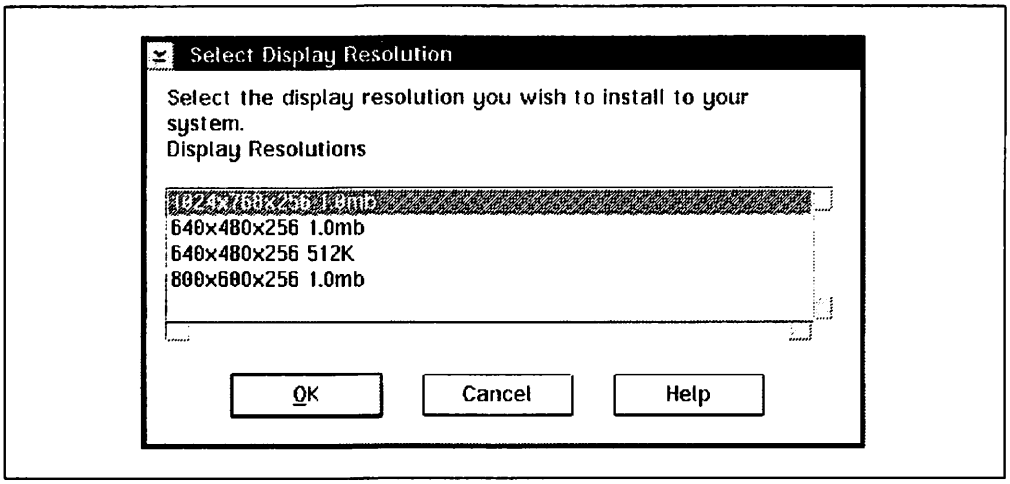


Figure 25. DSPINSTL - Select Display Resolutions

10. Select the list box item for the display resolution required.
11. Click on **OK**.

    The Source Directory window will then be displayed, as in Figure 26.

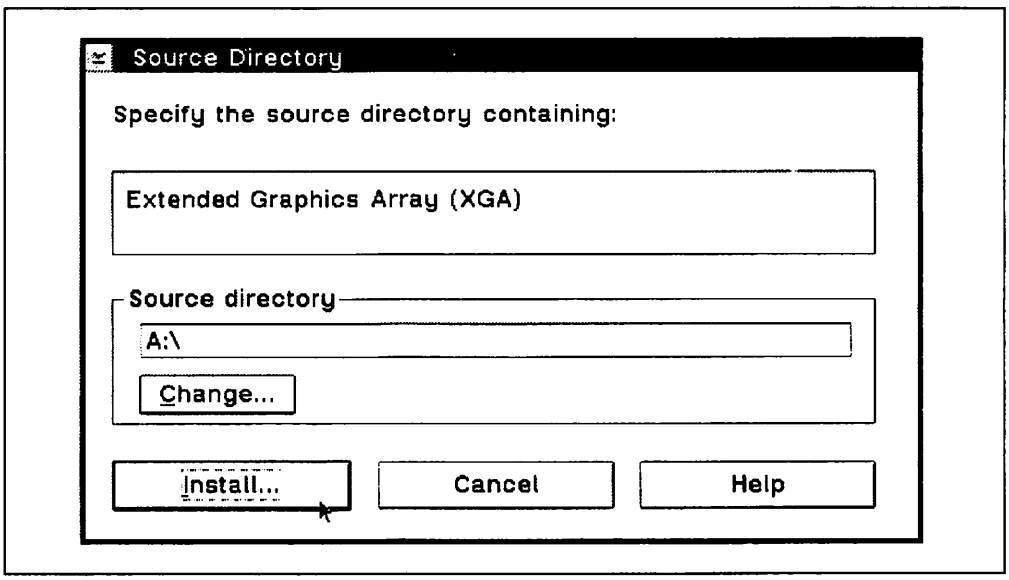


Figure 26. DSPINSTL - Source Directory

12. Either select the default directory, or change the pathname so it points to drive A:. You will need the display driver diskettes if you have specified the diskette drive as the path.
13. Click on **Install** and OS/2 will start the installation.

    When the installation has completed, the Display Driver Install message will be displayed, as in Figure 27 on page 34.

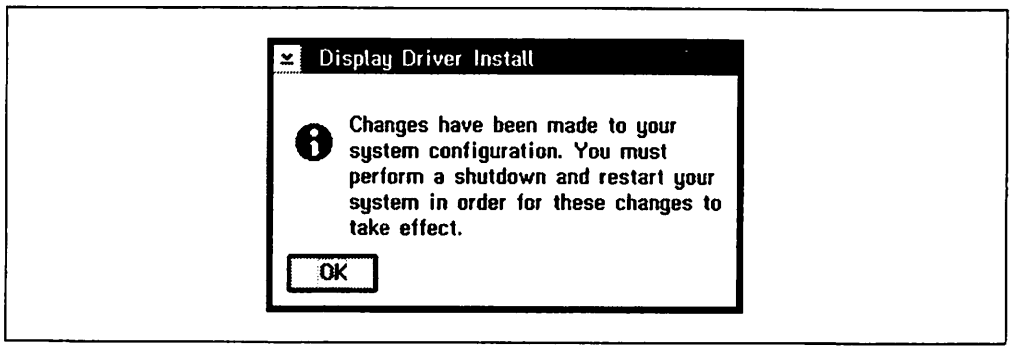


Figure 27. DSPINSTL - Display Driver Install Message

14. Click on **OK**.
15. To activate the changes, **Shut down** your system and restart it.

If at any time you want to change the SVGA display resolution, you must use this DSPINSTL procedure again.

This is because there are different SVGA display driver sets for each SVGA display resolution (unlike XGA, where one display driver set supports all resolutions).

Note

DSPINSTL does not handle the copying or installation of ISO fonts.

If you have an ISO compliant machine with an ISO adapter and an ISO display, and you want the ISO font support, then you should install the fonts again after running DSPINSTL.

In this case it would be normally faster and easier to use Selective Install for both display driver and ISO font installation.

3.4 OS/2 2.1 Installation on Loadable ABIOS Systems

As described in section 4.3, "Loadable ABIOS Support" on page 43, some of the more recent IBM PS/2 Micro Channel* systems load the ABIOS from disk into RAM, rather than using a copy of ABIOS in ROM.

The ABIOS files are supplied with the hardware, rather than as part of the OS/2 2.1 package. The ABIOS files are contained on the reference diskette, and are also preloaded onto the system partition (if one exists). The system partition is also known as the IML partition.

During the installation of OS/2 2.1 on a PS/2 system with loadable ABIOS, the installation procedure will first try to access the system partition on the fixed disk, using a special device driver which enables the system partition to be accessed as a standard drive. This device driver approach works for the PS/2 9556, 9557 and 9585.

If this is successful, the ABIOS files will be copied across to the OS/2 2.1 boot drive.

If the ABIOS files cannot be found, then the system will first try to find the ABIOS files on the fixed disk, and if they still cannot be found, then the user will be prompted for the reference diskette. This is sometimes also called the Hardware Support diskette.

Note

If the system did not come with a reference diskette, then you should create one following the instructions in the hardware manual. This has to be done before you start installation of OS/2 2.1.

3.5 Warning on ACL File Protection before Installation

If OS/2 2.1 is being installed over a previous version of OS/2, then the system checks whether any files are protected by ACL (Access Control List) before starting the installation process itself.

This check is important for OS/2 LAN Server Advanced systems which use the HPFS386 file system.

The ACL installation check uses a predefined list of directories and files to be checked, and then touches each file and directory using the DosSetFileMode call in order to test whether the file can be replaced.

If any files fail this test, then the OS/2 2.1 installation would also fail, and so the ACL Check produces a list of files protected by ACL and aborts the installation.

The user should then remove the ACL protection on these files and then restart the OS/2 2.1 installation.

This ACL Check approach is also used during Service Pak installation.

3.6 Configuring SVGA Display Drivers

The initial OS/2 2.1 installation of SVGA does not install full SVGA support, but only installs VGA support at a resolution of 640x480x16.

This is because there is no MVDM DOS support during the initial installation phase, and this is needed to run the DOS SVGA query program which establishes the capabilities of the display adapter and monitor.

To install full SVGA support, including the higher SVGA resolutions, the SVGA display drivers must be installed following the completion of the initial OS/2 2.1 installation. This can be done using either Selective Install or DSPINSTL.

Changing the SVGA resolution involves reinstallation of SVGA display drivers, since there is a different SVGA display driver set for each display resolution.

The high-resolution fonts are automatically installed, except for ISO fonts.

Warning

If you select a resolution not supported by your SVGA adapter/display combination, then your screen may be blank when OS/2 2.1 is rebooted. It is worth checking that the resolution is supported by the SVGA adapter/display combination before configuring it.

The SVGA.EXE program creates a file called SVGADATA.PMI, which contains information on the capabilities of the SVGA adapter. This program is now run automatically by the installation programs, and there is no need to explicitly run SVGA ON in OS/2 2.1.

Do not use the SVGA OFF command as this will erase the SVGADATA.PMI file, which is needed by the SVGA display driver support. Erasing this file will result in a blank screen appearing when the system is restarted.

3.7 Configuring XGA Display Drivers

XGA display driver installation is normally correct, since one set of XGA display drivers caters for all screen resolutions. In addition, the XGA-2 adapter is able to sense which display is attached using Display Mode Query Set (DMQS).

XGA needs the high-resolution fonts, which are installed automatically by the installation programs.

Changing the screen resolution is easily done using the Screen page which is added at the start of the Settings notebook of the System object. This is also sometimes referred to as Static Mode Switching (since the resolution can be changed without reinstallation of display drivers).

The DMQS procedure enables the XGA-2 adapter to find out the capabilities of the attached screen, such as resolutions supported, and thus the highest resolution can be selected automatically during installation.

Once the XGA drivers have been installed, you can easily change your XGA display resolution at any time using the **System Icon**, contained in the **System Setup** folder.

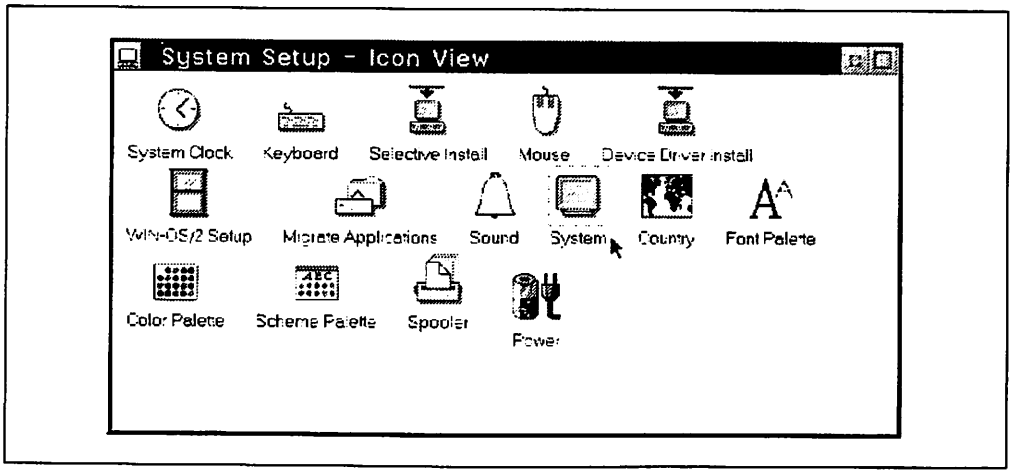


Figure 28. XGA Configuration - System Icon in System Setup Folder

The first panel of the System Icon Settings notebook is displayed, and this panel contains one page for XGA, and two pages for XGA-2.

The first page allows you to select the desired resolution, as shown in Figure 29 and Figure 30 on page 38.

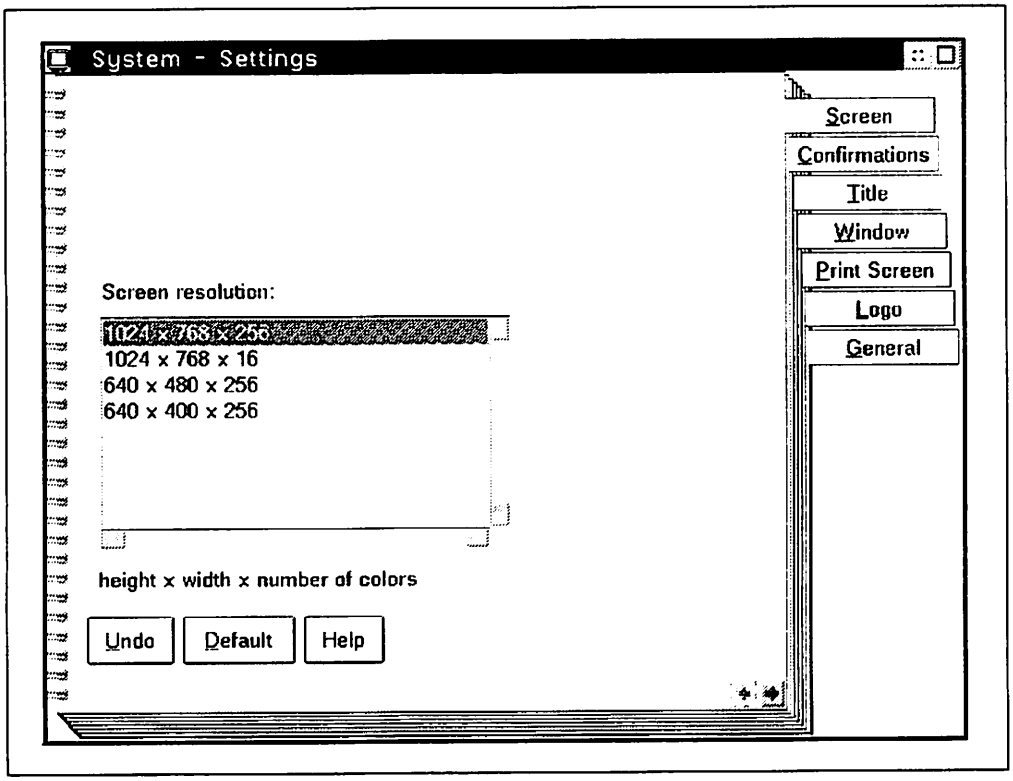


Figure 29. XGA Configuration - System Settings, XGA

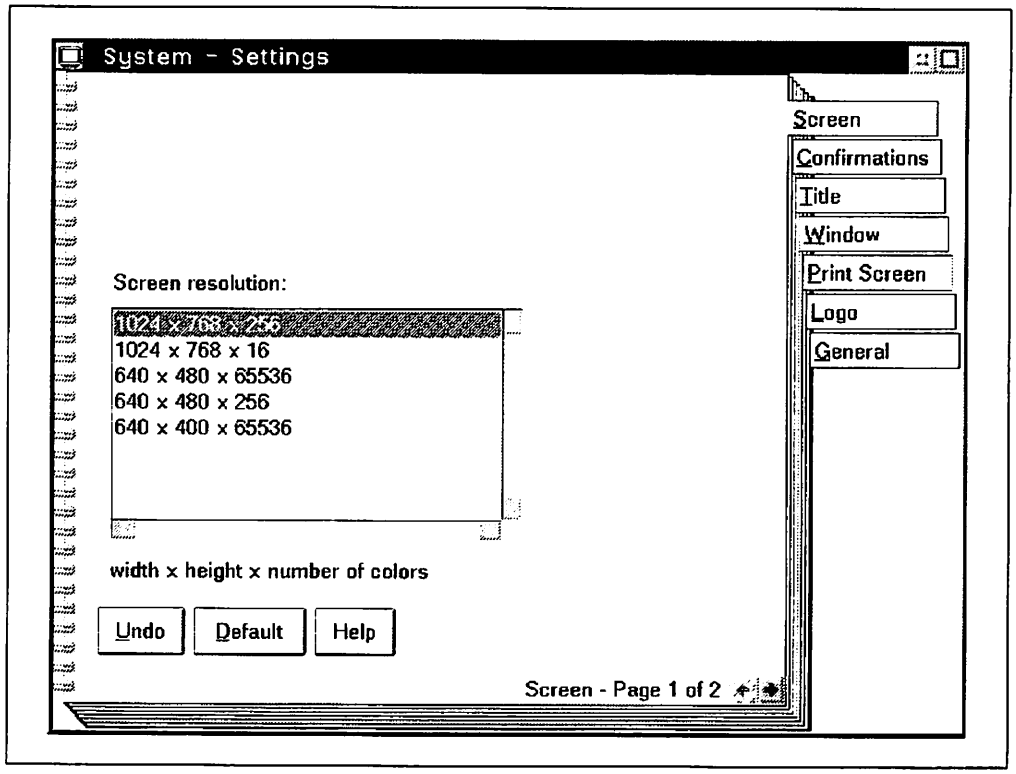


Figure 30. XGA Configuration - System Settings, XGA-2 First Page

The second page (on XGA-2 systems) allows you to define the type of screen attached to the system, as shown in Figure 31.

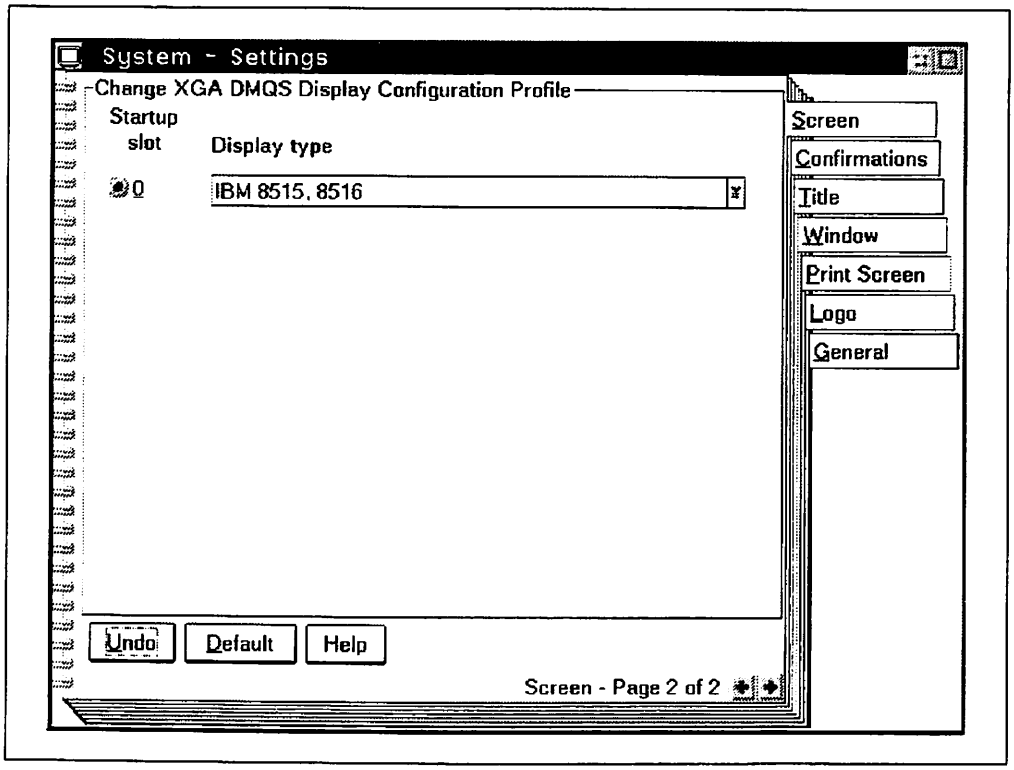


Figure 31. XGA Configuration - System Settings, XGA-2 Second Page

In most cases OS/2 can recognize the type of display attached and will input the corresponding .DGS driver, but sometimes the reply from the non-IBM display to the DMQS inquiry by the XGA adapter does not match the capabilities of the display. In this case, DMQS Override can be used to manually override the settings and thus OS/2 2.1 can exploit the full capabilities of the display.

This can be done by using the drop-down menu of possible displays, as shown in Figure 32.

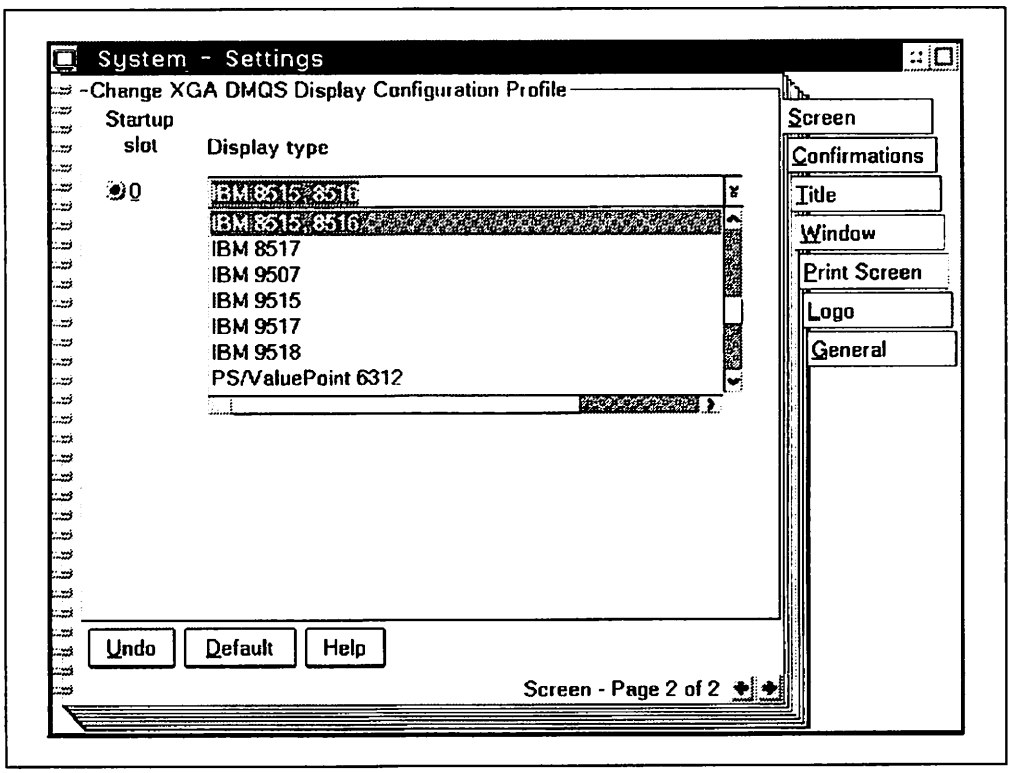


Figure 32. XGA Configuration - System Settings, XGA-2 DMQS Override

For more details on XGA-2 DMQS Override, see section 8.3.5, "DMQS and XGA-2" on page 111.

To activate the changes, **Shut down** your system and restart it.

3.8 Preloaded OS/2 2.1

OS/2 2.1 is preloaded onto selected systems by IBM and other PC manufacturers.

The preloaded system contains utilities for creating a set of bootable utility diskettes, removing OS/2 features, and configuring the system setup.

The utility diskettes which can be created are for supporting the preloaded system in case problems are encountered and the system cannot be booted normally. These diskettes include copies of the FDISK, FORMAT, CHKDSK, and BACKUP/RESTORE programs.

The utility for use in removing features allows the user to reclaim hard disk space by removing items from his system.

The configuration program allows the user to change the configuration support without inserting diskettes. Preloaded systems come with all features installed, and have extra directories which contain files needed for changing the configuration. No printer is configured (this is left for the user), and the systems are preconfigured for VGA display support.

These programs, along with a set of tutorials, service information, and a system information tool, can be found in a folder on the Desktop called **Preinstalled Essentials** (this used to be the Welcome Folder). In the box will be a book called "Using Your Preinstalled System", which explains how to use these programs and utilities.

Backup copies of the diskettes are supplied with some systems; the inclusion of backup diskettes varies between countries.

Chapter 4. Hardware Support and Enhancements

OS/2 2.0 provided an advanced 32-bit operating system for PCs based on the Intel 386 and 486 microprocessors. It was architected to be able to support and exploit a wide range of hardware, through the use of installable device drivers.

OS/2 2.1 extends that hardware support by including additional SCSI, SCSI-based CD-ROM and video device drivers in the OS/2 2.1 package. Additional device drivers will be available from the hardware device manufacturers and independent software developers, and also on bulletin boards.

In addition, OS/2 2.1 also includes changes to exploit Intel's new Pentium** chip, through the use of the Pentium virtual mode extensions.

4.1 Industry Hardware Support for OS/2 2.1

OS/2 2.1 has been tested on PC systems from a wide range of PC Manufacturers (PCMs), by the PCMs, by IBM at its OEM test laboratories in Boca Raton, Florida and Basingstoke, England, and also via two widespread beta tests of OS/2 2.1.

OS/2 2.1 has also been tested in conjunction with many PC adapters and peripherals from a wide range of OEM manufacturers and Independent Hardware Vendors (IHVs), by the OEMs and IHVs, by IBM, and via the beta tests.

The current list of IBM, PCM, OEM and IHV hardware supported by OS/2 2.1 is included in the **PCMTABLE** package, available on CompuServe, and to IBM internal users on OS2TOOLS.

The contents of these hardware support lists, as at the date of publication, has been reproduced in Appendix B, "OS/2 2.x Compatible PCM Systems" on page 305 and Appendix C, "OS/2 2.x Compatible Hardware Devices" on page 321.

If a PC system or hardware adapter or peripheral is not in these appendices, please check CompuServe for a more recent version of these lists, or contact the hardware manufacturer.

4.2 Understanding OS/2 Version 2 Hardware Support

OS/2 Version 2 supports a wide variety of hardware by insulating the operating system from the hardware. This is achieved through the use of installable device drivers and the BIOS firmware. New hardware devices can be supported by providing a new device driver, or by providing a standard BIOS interface on top of the new hardware.

This hardware abstraction layer has evolved many times since OS/2 1.0, and today this layer contains a variety of device driver and BIOS approaches. A simplified view of the hardware support components of OS/2 2.1 is shown in Figure 33 on page 42.

Hardware support in OS/2 2.1 is provided by a combination of the BIOS and device drivers. Most of this also applies to OS/2 2.0.

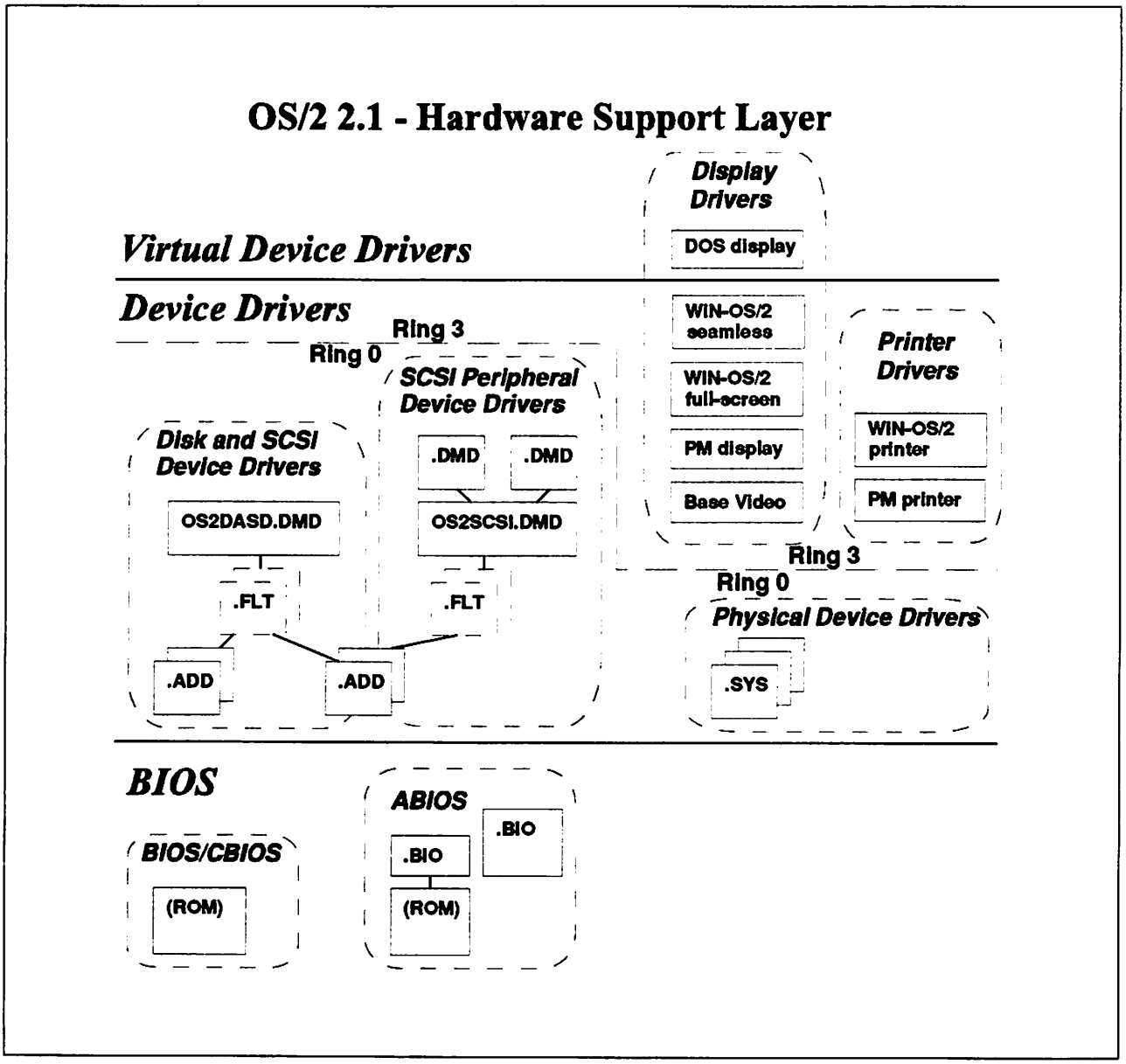


Figure 33. Overview of OS/2 2.1 Hardware Support Layer

4.2.1 BIOS

The BIOS provides the lowest level of the OS/2 2.1 operating system, and is normally contained in ROM on the system board. Although the BIOS is software, it is provided by the hardware manufacturer along with the PC.

The ROM BIOS is assigned the address range F000-FFFF (E000-FFFF on PS/2s). Originally introduced with the DOS operating system and the first IBM PC, the BIOS provides a standard set of basic functions for input/output devices such as the keyboard and the disks.

All IBM PCs and compatible systems from PC manufacturers ("clones") contain a BIOS. Although many DOS applications avoid using the BIOS for some functions (especially video), it is normally safe to assume that the BIOS is present. This BIOS is referred to as CBIOS (for Compatibility BIOS) on IBM PS/2s.

The IBM PS/2 Micro Channel systems also contain an ABIOS (for Advanced BIOS). This was designed to provide a BIOS optimized for use by multitasking operating systems such as OS/2. ABIOS is also normally held in ROM and mapped into the operating system address space (between 640KB and 1MB), but in some recent PS/2 models the ABIOS is held instead on disk and loaded into RAM (see 4.3, "Loadable ABIOS Support" for more details).

OS/2 2.1 uses the ABIOS if present; otherwise it will use the BIOS. For some input/output functions, OS/2 2.1 also accesses the hardware directly (through the device drivers).

4.2.2 Device Drivers

The other layer of software insulating the kernel of OS/2 2.1 from the hardware is the device driver layer.

Device drivers were originally provided as *.SYS files, typically one for each hardware component, which were loaded during the operating system initialization.

There are now a variety of device driver types and flavors in OS/2 2.1. Disk and SCSI device drivers have now adopted a layered device driver structure, which reduces the amount of effort needed to support a new hardware device. Video and printer drivers also have a specialized module structure.

Most device drivers are specified in the CONFIG.SYS file, using the BASEDEV= or DEVICE= keywords. The BASEDEV device drivers are needed for OS/2 2.1 to run and are loaded first, followed by the DEVICE driver drivers.

In addition to these device drivers, sometimes called physical device drivers, there are corresponding Virtual Device Drivers (VDDs) which provide device support for DOS and Windows applications running in the MVDM and WIN-OS/2 environments. VDDs interface to the physical device drivers, rather than accessing the hardware directly.

4.3 Loadable ABIOS Support

ABIOS is implemented on IBM PS/2 Micro Channel systems, and provides an additional BIOS optimized for use by a multitasking operating system such as OS/2.

The original ABIOS implementations were in ROM, and thus ABIOS could be assumed to be always present (in the same way that DOS assumes that the BIOS is always present).

When changes needed to be made to the ABIOS, they were implemented as ABIOS patch files (*.BIO files). These were listed in the ABIOS.SYS file. ABIOS patch files included both general patch files, and patch files for specific PS/2 models.

The ROM ABIOS implementation on IBM PS/2s has stored the ABIOS code in the same 128KB ROM as the POST (Power-On Self Test diagnostics) and the BIOS (CBIOS) code.

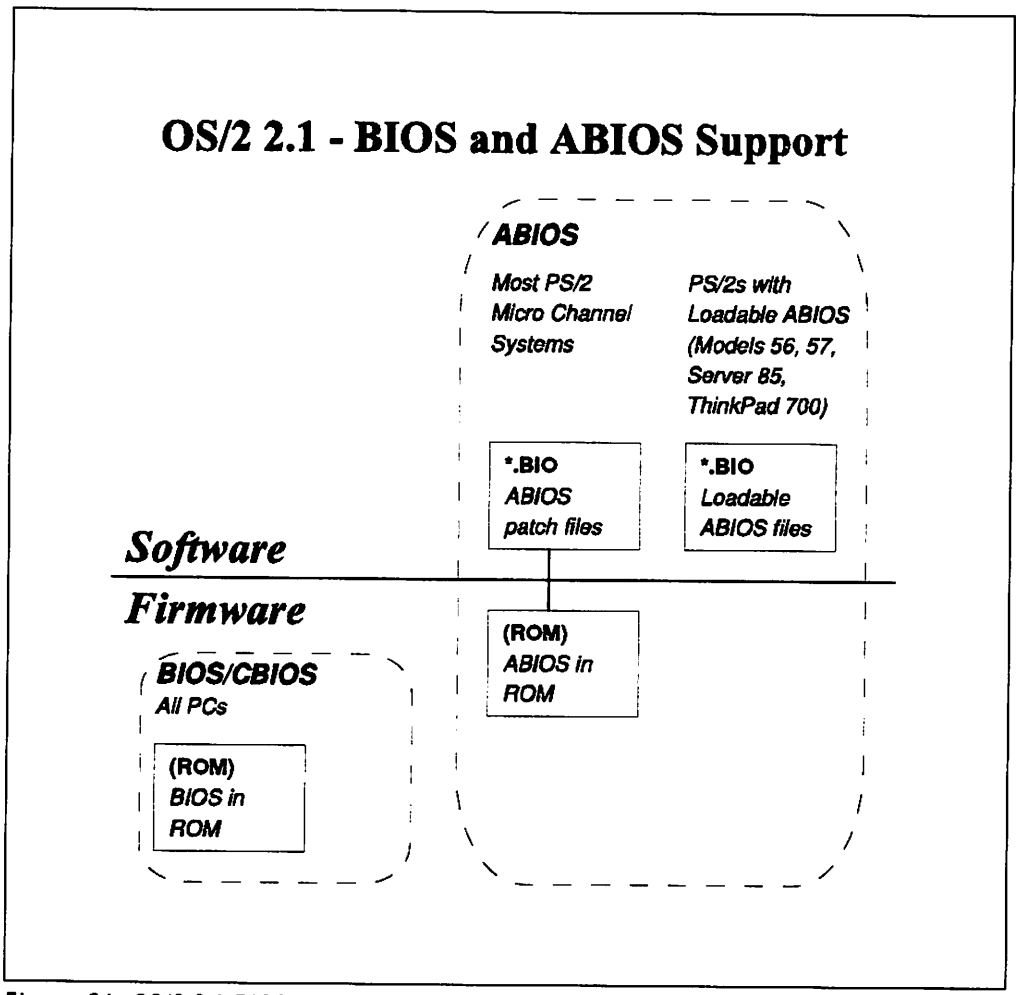


Figure 34. OS/2 2.1 BIOS and ABIOS Support

As PS/2 systems have become more advanced, some systems have literally run out of room in the 128KB ROM to store all of POST, BIOS and ABIOS. It has thus been necessary to move the ABIOS out of the ROM and into a file on the disk which can be loaded into RAM as part of the operating system. This is known as "loadable ABIOS".

Loadable ABIOS has been implemented on the following systems:

- PS/2 Models 56 and 57
- PS/2 Server 85
- ThinkPads 700, 700C, 720 and 720C

The introduction of loadable ABIOS has implications for OS/2 Version 2. Both OS/2 2.0 and OS/2 2.1 depend on the presence of ABIOS to run on IBM PS/2 systems (although they both run on IBM and PCM clones using a mixture of BIOS and direct hardware support).

Hence OS/2 Version 2 needs to

- Know which ABIOS files to load
- Install the appropriate ABIOS files during OS/2 Version 2 installation

The ABIOS files are specified in the ABIOS.SYS file, in the same way as ABIOS patch files are specified. However, in this case it is the complete ABIOS which is being loaded.

Loadable ABIOS is supported by OS/2 2.1, OS/2 2.00.1, and OS/2 2.0 with Service Pak XR06055 applied.

The OS/2 2.1 installation procedure has been enhanced to install the appropriate ABIOS file on the fixed disk. The ABIOS file is either copied from the system partition using a special device driver, or if this is not possible, the user is prompted for the "Hardware Support Diskette". At this stage, the PS/2 reference diskette should be inserted. The ABIOS.SYS file is then modified to include the ABIOS filename as its first entry. See 3.4, "OS/2 2.1 Installation on Loadable ABIOS Systems" on page 34 for more details.

4.4 Basic Hardware Support

The following additions have been made to the basic hardware support included with OS/2 2.1:

- Support for the PS/2 Server 195 and 295
- Support for the new Brazilian keyboard
- Support for the enhanced features of the new 2.88MB diskette drives
- Support for the new 3.5" enhanced rewritable optical disk drive

In addition, the following enhancements have been made to support laptop computers:

- Advancd Power Management (APM) support
- PCMCIA support
- VGA large cursor support on LCD screens
- Support for the Trackpoint II pointing device

The laptop enhancements are discussed in more detail in Chapter 11, "Enhancements for Laptops and Notebooks" on page 147.

4.4.1 PS/2 Server 195 and 295 Support

The PS/2 Server 295 is a high-performance, high-availability system, designed for use as an advanced database and application server. The Server 295 has been designed as a server which is based on PC technology, such as the Intel 486 microprocessor and the Micro Channel, but which also introduces new technology such as multiple processors, a fault-tolerant disk subsystem, and remote monitoring and control. The PS/2 Server 195 is an entry-level version of the Server 295.

When the Server 295 was announced, software support was only provided for a modified version of OS/2 1.3, which included its own memory management extensions and disk device drivers. In addition, extensions were provided to enable asymmetric multiprocessing in conjunction with OS/2 LAN Server 2.0 Advanced.

OS/2 2.1 includes changes to support the Server 295 disk architecture, including interfacing with the firmware for Orthogonal RAID-5 fault-tolerance support, and to remap the SCSI disk buffers from virtual mode into real mode so that MVDM sessions are able to access the disks.

A new version of MultiProcessing Extensions/2 will also be provided for use in conjunction with OS/2 2.1 and OS/2 LAN Server 3.0. This continues the support for asymmetric multiprocessing provided on an OS.2 1.3 base.

Asymmetric multiprocessing is a form of multiprocessing where each processor runs a specific set of software (whereas symmetric multiprocessing, as providing in the Mach microkernel, distributes tasks dynamically between processors). In the case of the Server 295, OS/2 runs on both processors, and then one processor (the File Processor) runs system functions such as the HPFS386 file system and the network drivers, and the other processor (the Application Processor) runs application functions such as a database manager.

For more details about the PS/2 Server 295 hardware, see the *Advanced PS/2 Servers - Planning and Selection Guide*, GG24-3927.

4.4.2 Brazilian Keyboard Support

A new national keyboard has been introduced by the Brazilian government for the Portugese language, and support for this keyboard is provided in OS/2 2.1.

The new Brazilian keyboard is similar to the standard PS/2 enhanced keyboard, but with a few extra keys - the Brazilian keyboard has 104 keys. Use of the accented and dead-key operation is the same as for other keyboards.

4.4.3 Enhanced 2.88MB Diskette Drive Support

There are three varieties of PS/2 2.88MB diskette drive:

- 2.88MB diskette drive (6451106)
- Enhanced 2.88MB diskette drive with software eject (6451272)
- Enhanced 2.88MB diskette drive with software eject and software lock/unlock (6451271)

OS/2 2.1 includes support for all three drives, including the enhanced features for software eject and software lock/unlock where appropriate.

Software eject enables the diskette to be ejected under software control, typically from the pop-up menu of the diskette icon. This is a usability feature.

Software lock/unlock enables the diskette drive to be locked or unlocked under software control, again typically from the pop-up menu of the diskette icon. This provides additional security protection for the workstation and its data.

4.4.4 3.5" Enhanced Rewritable Optical Drive Support

IBM's new 3.5" Enhanced Rewritable Optical Drive (6451295) supports P-ROM optical disks, as well as the MO and O-ROM optical disks supported by the previous 3.5" Rewritable Optical Drive.

All three optical disks are provided in 3.5" cartridges. The differences are:

- MO (magneto optical) disks can be read and written many times in the optical drives. MO disks are usually formatted at 127MB.
- O-ROM (optical read-only memory) disks can only be read in optical drives; they must be created using special equipment (as with CD-ROMs). O-ROM disks are usually formatted at 122MB.
- P-ROM (partial read-only memory) disks are a hybrid of MO and O-ROM disks, and contain a designated read-only area and a rewritable area. The

whole P-ROM disk can be read and the rewritable portion written using the Enhanced Rewritable Optical Drive. P-ROM disks are usually formatted at 122MB.

The OS/2 format utility has been enhanced to be able to format P-ROM disks in the Enhanced Rewritable Optical Drive. See section 5.3, "Format Utility Support for P-ROM Optical Disks" on page 64 for more details.

The 3.5" Enhanced Rewritable Optical Drive also includes software eject and software lock/unlock features, and these can be used from OS/2 2.1, typically from the pop-up menu of the optical drive icon.

4.5 Device Driver Support

The basic hardware device drivers are provided as *.SYS files. Apart from the keyboard, screen and clock device drivers (which are loaded automatically), these device drivers are specified in the CONFIG.SYS file.

Device drivers are included in the CONFIG.SYS file either with the BASEDEV= statement or the DEVICE= statement.

BASEDEV device drivers are loaded first, as they are needed in order to load the rest of the operating system. Since they are loaded early, no path can be specified, and the device driver files must be in either the root directory or the \OS2 directory of the boot drive.

DEVICE device drivers are then loaded, and a pathname can be specified.

For a more comprehensive discussion of device driver architectures, along with sample code, refer to the *IBM OS/2 Device Driver Development Kit* which is available on CD-ROM.

4.5.1 Physical Device Driver Support

The following basic device drivers are automatically loaded by OS/2 2.1 and do not need to be specified in the CONFIG.SYS file:

Table 11. Basic Device Drivers Automatically Loaded in OS/2 2.1

Device	ISA device driver	Micro Channel device driver
Clock	CLOCK01.SYS	CLOCK02.SYS
Keyboard	KBD01.SYS	KBD02.SYS
Screen	SCREEN01.SYS	SCREEN02.SYS

The following basic device drivers need to be specified explicitly in the CONFIG.SYS file, using the BASEDEV= statement:

Table 12. Basic Device Drivers in OS/2 2.1 Loaded with BASEDEV=

Device	ISA device driver	Micro Channel device driver
Printer	PRINT01.SYS	PRINT02.SYS

The following device drivers (used for both ISA and Micro Channel systems) are included with OS/2 2.1 and should be specified if necessary in the CONFIG.SYS file using the DEVICE= keyword:

Table 13. Basic Device Drivers in OS/2 2.1 Loaded with DEVICE=

Device	Device Driver
Hardware Configuration Testing (used by hardware presence check programs)	TESTCFG.SYS
Presentation Manager Draw Support	PMDD.SYS
Mouse Pointer Draw Support	POINTDD.SYS
Mouse Support	MOUSE.SYS
Touch Devices	TOUCH.SYS
System Error Logging	LOG.SYS
Serial Device Support	COM.SYS
PCMCIA Bus Support	PCMCIA.SYS
Advanced Power Management Support	APM.SYS
External Diskette Support	EXTDSKDD.SYS

4.6 Disk and SCSI Adapter Support

OS/2 2.1 provides support for a number of SCSI and non-SCSI disk adapters, for both Micro Channel and ISA PCs.

This support is provided through a combination of the hardware-specific SCSI adapter device drivers (.ADDs), and the hardware-independent disk device manager (OS2DASD.DMD). In addition, filter device drivers can be installed between the .ADDs and the .DMD to provide specialist function such as encryption.

This layered device driver approach enables new SCSI hardware adapters to be supported with minimum effort, simply by writing a new adapter device driver (.ADD file).

At installation time, OS/2 can in most cases sense the installed adapters and will then load the necessary .ADD files and include the necessary BASEDEV statements in the CONFIG.SYS.

If a suitable device driver cannot be found for all the disks installed in the system, then the IBMINT13.I13 device driver is installed. This uses the BIOS to provide the disk support.

Lists of the disk and SCSI adapters supported by OS/2 2.1 are provided in 4.6.2, "Disk and SCSI Adapter Support in OS/2 2.1" on page 49.

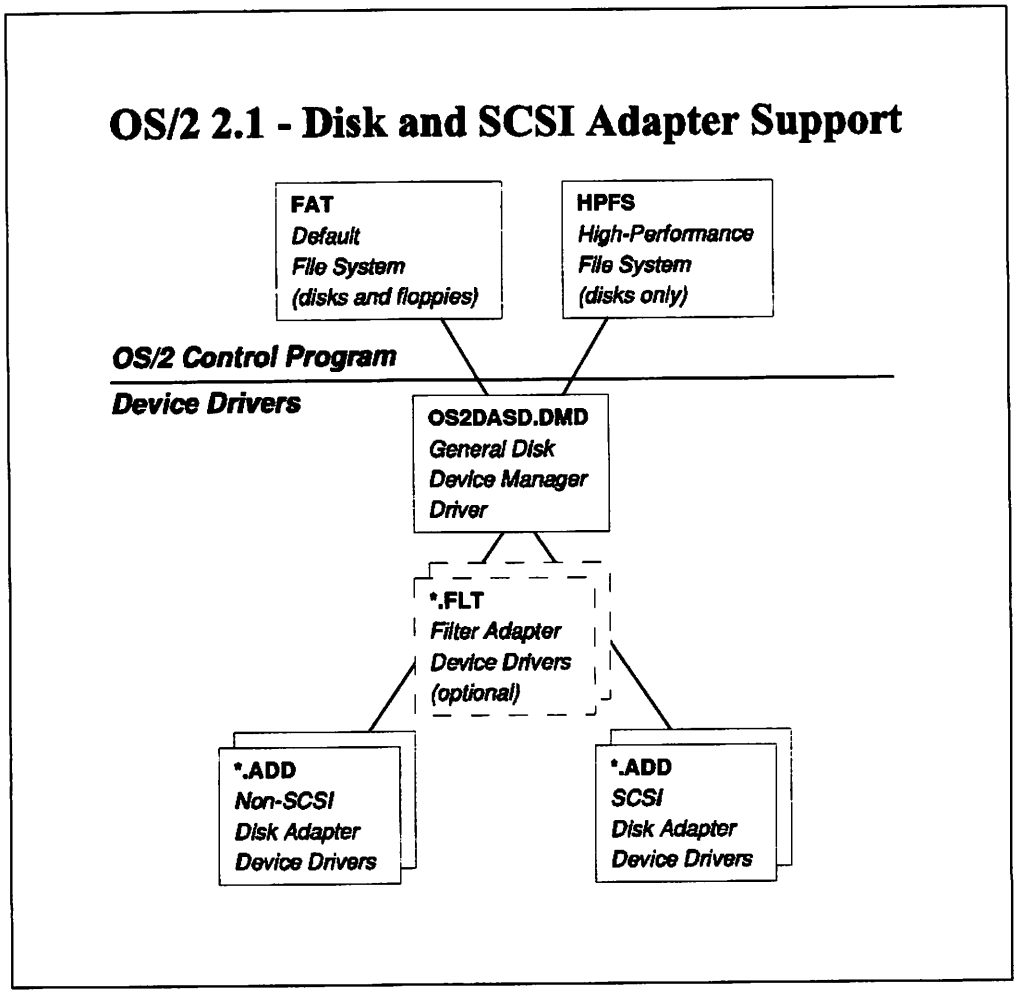


Figure 35. OS/2 2.1 Disk and SCSI Adapter Support

4.6.1 OS/2 2.1 Enhancements

OS/2 2.0 included SCSI adapter support for the IBM PS/2 Micro Channel SCSI adapters, and also default support for ISA SCSI adapters using the INT13 BIOS interface.

OS/2 2.1 adds the leading SCSI adapters, including Adaptec, DPT and Future Domain, as well as IBM.

4.6.2 Disk and SCSI Adapter Support in OS/2 2.1

To find out which SCSI adapter device drivers are supplied with OS/2 2.1, look at the SCSI.TBL file in the \OS2\INSTALL directory. This list is used by the Selective Install program, and contains a list of the SCSI adapters along with the corresponding device drivers, and also the name of the appropriate hardware presence check program.

Table 14. Disk Adapter Device Drivers Supplied with OS/2 2.1		
Device	**ISA device driver**	**Micro Channel device driver**
Diskette	IBM1FLPY.ADD	IBM2FLPY.ADD
Non-SCSI Disks	IBM1S506.ADD	IBM2ADSK.ADD
PS/2 Model 57 Disk support		IBM2M57.ADD

Table 15. SCSI Adapter Device Drivers Supplied with OS/2 2.1	
Adapter	**Device Driver**
Adaptec A/C 6260, AHA-1510, 1520, 1522	AHA152X.ADD
Adaptec AHA-1540, 1542	AHA154X.ADD
Adaptec AHA-1640	AHA164X.ADD
Adaptec AHA-1740, 1742, 1744	AHA174X.ADD
DPT PM-2011, PM-2012	DPT20XX.ADD
Future Domain TMC-845, 850, 850IBM, 860, 875, 885	FD8XX.ADD
Future Domain TMC-1650, 1660, 1670, 1680, MCS-600, 700	FD16-700.ADD
Future Domain FD7000EX	FD7000EX.ADD
IBM 16-bit AT Fast SCSI Adapter	FD16-700.ADD
IBM PS/2 SCSI Adapter and SCSI Adapter with Cache	IBM2SCSI.ADD
Default General ISA SCSI Adapter Support	IBMINT13.I13

4.7 SCSI-based CD-ROM and Other SCSI Device Support

OS/2 2.1 supports SCSI-based CD-ROM drives and other SCSI-attached devices, such as read/write optical drives.

This support is provided through a combination of the hardware-specific SCSI adapter device driver (.ADD), the hardware-independent SCSI device manager (OS2SCSI.DMD), and a device-specific SCSI peripheral device driver (OS2CDROM.DMD or similar). In addition, a filter device driver can be installed, and a typical use of this is to convert SCSI-2 commands issued by the OS2CDROM.DMD driver into SCSI-I commands which can be understood by more CD-ROM devices.

This layered device driver approach enables new SCSI CD-ROM drives and other SCSI peripheral devices to be supported with minimum effort.

The CD-ROM file system support implemented in OS/2 2.1 with CDFS.IFS includes support for CD-XA. OS/2 and DOS applications can thus take advantage of CD-XA. DOS support is provided in the MVDMs through the VCDROM.SYS virtual device driver. This implementation of CD-XA support in VCDROM is independent of the MSCDEX enhancements.

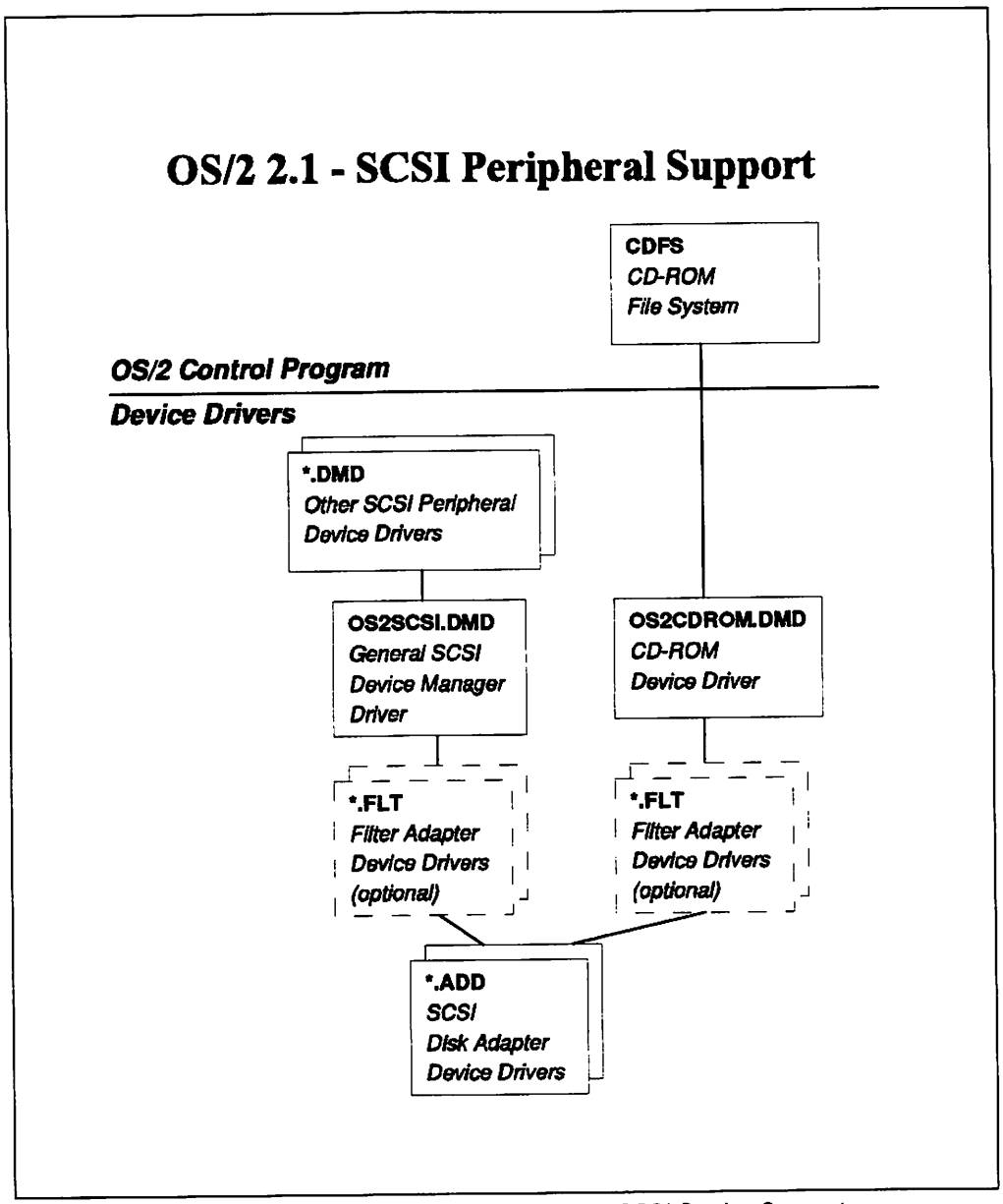


Figure 36. OS/2 2.1 SCSI-attached CD-ROM and Other SCSI Device Support

At installation time, OS/2 can in most cases sense the installed CD-ROM drive and will then load the necessary .DMD and .FLT files and include the necessary BASEDEV and DEVICE statements in the CONFIG.SYS.

OS/2 2.1 does not currently include as standard device drivers for proprietary (non-SCSI-attached) CD-ROM drives. However, IBM is working with CD-ROM drive manufacturers to develop these, and because of the pluggable design of OS/2 2.1, these CD-ROM device drivers could be distributed via bulletin boards and easily installed.

Details of the SCSI-attached CD-ROM devices supported by OS/2 2.1 are provided in 4.7.2, "SCSI-Attached CD-ROM Support in OS/2 2.1" on page 52.

4.7.1 OS/2 2.1 Enhancements

OS/2 2.0 included CD-ROM support for the IBM CD-ROM drive.

OS/2 2.1 adds CD-ROM support for a range of CD-Technology, Hitachi**, NEC**, Panasonic**, Sony**, Texel and Toshiba** CD-ROM drives.

Some of these drives require an additional filter driver in order for them to work correctly in OS/2 2.1. These filter drivers are required to support CD-ROM drives which adhere to the SCSI-1 standard, instead of the new SCSI-2 standard. The new filter drivers are used to convert the SCSI-2 commands generated by the OS2CDROM.DMD module into the vendor-unique SCSI-1 commands required by the SCSI-1 CD-ROM drive.

4.7.2 SCSI-Attached CD-ROM Support in OS/2 2.1

Table 16 lists the CD-ROM devices supported by OS/2 2.1, along with the corresponding device drivers and filters. file in the \OS2\INSTALL directory. This list is used by the Selective Install program, and contains a list of the CD-ROM drives along with the name of the corresponding device driver, and filter device driver where needed.

For a list of SCSI and CD-ROM Drive restrictions, and for information on Kodak** MultiSession Photo-CD support, refer to *OS/2 2.1 Using the Operating System.*

Table 16. CD-ROM Device Drivers Supplied with OS/2 2.1

CD-ROM Drive	Device Driver	Filter (if needed)
CD Technology T3301	OS2CDROM.DMD	
Hitachi CDR-1650S, 1750S, 3650	OS2CDROM.DMD	HITCDS1.FLT
Hitachi CDR-3750	OS2CDROM.DMD	
IBM CD-ROM I	OS2CDROM.DMD	TOSHCDS1.FLT
IBM CD-ROM II	OS2CDROM.DMD	
NEC Intersect CDR-25, 36, 37, 72, 73, 74, 82, 83, 84	OS2CDROM.DMD	NECCDS1.FLT
NEC MultiSpin CDR-38, 74, 84	OS2CDROM.DMD	
Panasonic CR-501, LK-MC501S, MC501B, MC521	OS2CDROM.DMD	
Pioneer DRM-600, DRM-604X	OS2CDROM.DMD	
Sony CDU-541, 561, 6211, 7211	OS2CDROM.DMD	
Sony CDU-6111	OS2CDROM.DMD	SONYCDS1.FLT
Texel DM-3021, 5021	OS2CDROM.DMD	SONYCDS1.FLT
Texel DM-3024, 5024	OS2CDROM.DMD	
Toshiba 3201	OS2CDROM.DMD	TOSHCDS1.FLT
Toshiba 3301, 3401	OS2CDROM.DMD	

This list is also held in the file CDROM.TBL held in the \OS2\INSTALL directory.

4.8 Advanced SCSI Programming Interface

Support is also provided for the Advanced SCSI Programming Interface (ASPI) developed by Adaptec.

This is provided through OS2ASPI.DMD, which is an ASPI device manager driver that is compatible with existing device modules written to Adaptec's ASPI specification.

For more details, refer to the *IBM OS/2 Device Driver Development Kit.*

4.9 Video Support

OS/2 2.0 was designed to be able to exploit a wide range of display adapters, This support is provided through a range of DLLs and device drivers, even for a single display adapter. This range is necessary in order to provide video support for a wide range of applications including DOS, WIndows and OS/2 character-mode applications as well as the native Presentation Manager applications.

4.9.1 OS/2 2.1 Enhancements

OS/2 2.1 includes the new 32-bit PM graphics engine, and new 32-bit display device drivers for XGA, XGA-2, SVGA, 8514/A, and VGA display adapters.

The SVGA, XGA, XGA-2 and 8514/A display drivers support sessions running in the background and foreground, in full-screen and seamless modes, at resolutions of up to 1024 x 768, with support for up to 256 colors.

Because of its design, OS/2 2.1 is enabled for new display device drivers to be added as they become available from IBM or third parties. IBM is also working with the manufacturers of other leading display adapters and independent software vendors to ensure the accelerated display functions, such as those implemented on the S3 805 and 926 chipsets, are exploited under OS/2 2.1. These display drivers will be distributed on bulletin boards as they become available.

The SVGA display driver supports a range of SVGA adapters using the following chipsets:

- IBM VGA 256-color
- Cirrus Logic
- Tseng ET4000
- Western Digital Imaging WD90C11, C30, C31 (C30 mode only)
- Trident Microsystems TVGA8900
- ATI 28800
- Headland Technology HT209

The IBM VGA 256-color chipset is only supported in 640 x 480 x 256 colors mode; all the other SVGA chipsets are supported in the following modes:

- 640 x 480 x 256 colors
- 800 x 600 x 256 colors
- 1024 x 768 x 256 colors

New 32-bit display drivers are also provided for VGA adapters (in 640 x 480 x 16 color mode), and XGA adapters (in a range of resolutions up to 1024 x 768 x 256 color mode).

The old 16-bit display drivers (such as those for CGA or EGA) will still work in conjunction with the 32-bit graphics engine, but may not provide seamless Windows support or as good performance as 32-bit display drivers.

Note

The new 32-bit display drivers cannot be used with the old 16-bit PM graphics engine, which is used in OS/2 1.x or OS/2 2.0.

A list and details of the display adapters supported by OS/2 2.1 are provided in 4.9.2, "Video System Support in OS/2 2.1."

4.9.2 Video System Support in OS/2 2.1

To find out which display drivers are supplied with OS/2 2.1, examine the \OS2\INSTALL directory and list the files with extension .DSC. These are the display configuration files. Each of these display configuration files contains information used by the utility DSPINSTL.EXE to install the video adapter support.

Table 17 (Page 1 of 2). Display Drivers Supplied with OS/2 2.1

Mode	Chipset	Resolutions	Display Driver Type	WIN-OS/2 Support
CGA		320x200x4	16-bit	Full-screen
EGA		640x350x16	16-bit	Full-screen
VGA		640x480x16	32-bit	Seamless, Full-screen
8514	8514/A	1024x768x16	32-bit	Seamless, Full-screen
SVGA	IBM VGA 256-color	640x480x256(512KB) 640x480x256(1MB)	32-bit	Seamless, Full-screen
SVGA	Tseng ET4000	640x480x256(512KB) 640x480x256(1MB) 800x600x256(1MB) 1024x768x256(1MB)	32-bit	Seamless, Full-screen
SVGA	ATI 28800	640x480x256(512KB) 640x480x256(1MB) 800x600x256(1MB) 1024x768x256(1MB)	32-bit	Seamless, Full-screen
SVGA	Cirrus CL-GD5422, CL-GD5424	640x480x256(512KB) 640x480x256(1MB) 800x600x256(1MB) 1024x768x256(1MB)	32-bit	Seamless, Full-screen
SVGA	Headland HT209	640x480x256(512KB) 640x480x256(1MB) 800x600x256(1MB) 1024x768x256(1MB)	32-bit	Seamless, Full-screen

Table 17 (Page 2 of 2). Display Drivers Supplied with OS/2 2.1				
Mode	**Chipset**	**Resolutions**	**Display Driver Type**	**WIN-OS/2 Support**
SVGA	Western Digital WD90C11, WD90C30, WD90C31	640x480x256(512KB) 640x480x256(1MB) 800x600x256(1MB) 1024x768x256(1MB)	32-bit	Seamless, Full-screen
SVGA	Trident TVGA8900B, TVGA8900C	640x480x256(512KB) 640x480x256(1MB) 800x600x256(1MB) 1024x768x256(1MB)	32-bit	Seamless, Full-screen
XGA	XGA, XGA-2	1024x768x256 1024x768x16 640x480x256 640x400x256	32-bit	Seamless, Full-screen

4.9.3 Understanding Video Adapters

Some of the toughest configuration issues revolve around configuring an optimal video solution for OS/2. This section is designed to describe the issues involved in video configuration, and ways to navigate sucessfully through them.

In this section:

- **Video adapter** refers to the video subsystem of a computer on a board in the bus, or designed into the motherboard; for example the Orchid Prodesigner IIs or Catseye XGA-2.
- **Video chipset** refers to a family of chips from one manufacturer which are backward compatible with each other, and have defining characteristics; for example Trident 8900A, 8900B, 8900C, or Chips and Technologies 82c451, 82c452, 82c453.

4.9.3.1 Background and History

Video is possibly the fastest changing area of the Personal Computer marketplace. Currently, one or more new video chipsets are being introduced each month, along with many boards based on new and old chipsets. One or two of these boards might have the right combination of price, function, and marketing to capture a major portion of the market (5% - 10%). Figure 37 on page 56 shows a graphical representation of some of the competing standards which have been introduced in the last 15 years, and illustrates the lack of standards today in the video marketplace.

This lack of video standards has resulted in the need for new display drivers to be written for each new video adapter that appears. Typically, a manufacturer will release a board once display drivers have been ready for the applications or environments with the main market share; in the past this has meant DOS (BIOS), Windows, and possibly Autocad. Drivers for older versions of Windows or older revisons of their boards are often not available.

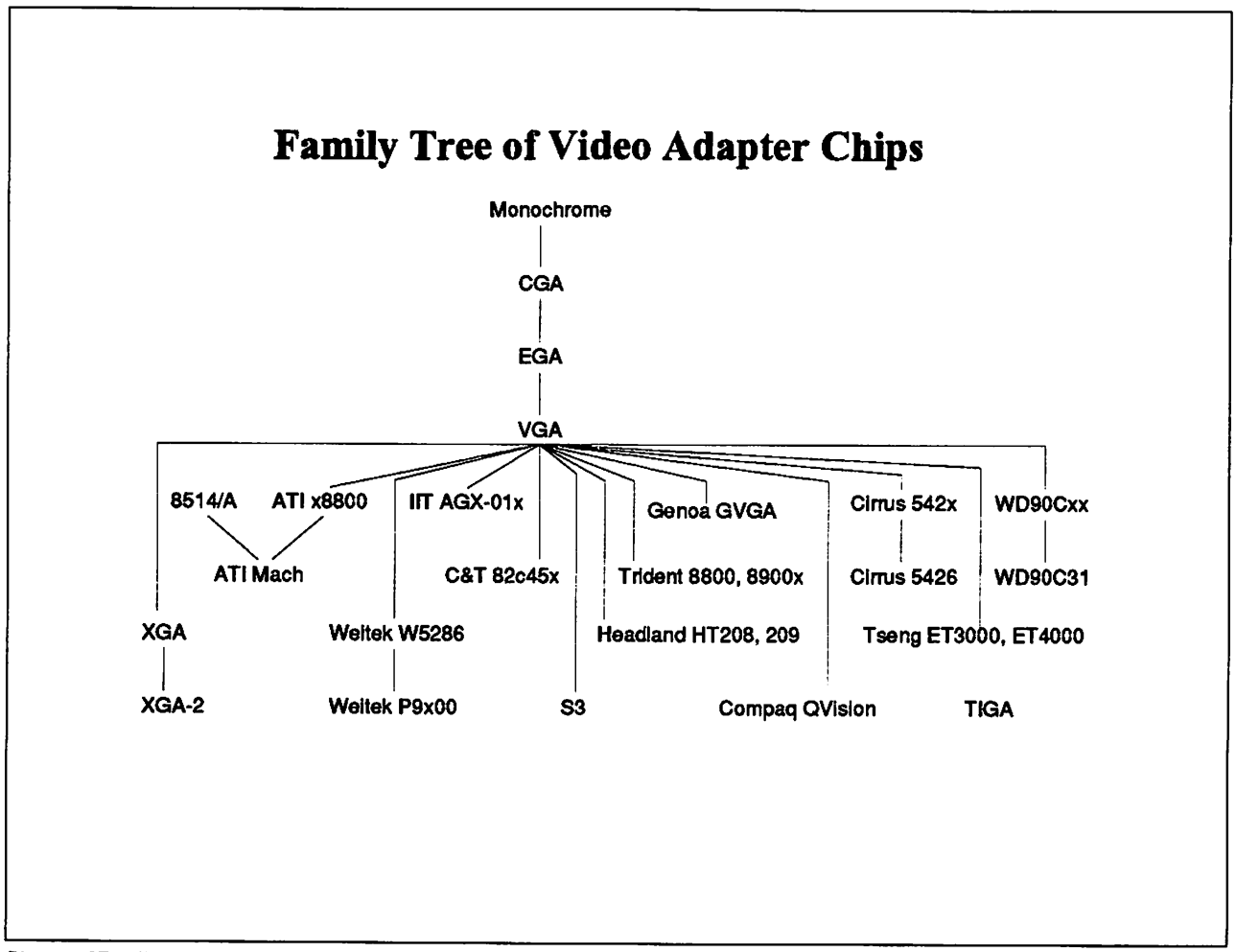


Figure 37. Video Chipset Family Tree

For a new operating system like OS/2 2.x, this has posed significant challeges. Typically, older boards may not be available for sale anymore, but may have a large percentage of the installed base. In addition, developing quality display drivers is a long process. In order to support boards which are popular, forecasts must be made to guess which boards will be at their market peak 6 to 8 months later, when the driver development process is complete.

4.9.3.2 Video Adapter Support and OS/2

OS/2 is using a three part approach to supporting the wide range of video boards:

1. The core video standards are supported (CGA, EGA, VGA, 8514, and XGA)
2. Generic drivers for a large number of popular SVGA chipsets have been developed (Tseng, ATI, Cirrus, Trident, WD, Headland). Manufacturers can then take the source code to these generic drivers, and produce drivers which are optimized for their individual boards.

   The full source code to all of the device drivers shipped with OS/2 2.x is available on CD-ROM for a nominal cost from the OS/2 Worldwide Developer's assistance program, or by entering GO OS2DAP in Compuserve.
3. For newer chipsets like S3, IIT AGX, and Weitek P9000, OS/2 drivers are being developed by the manufacturers. S3 has developed a set of OS/2 drivers which support their boards, and source code for 8514 and XGA

display drivers is available for the manufacturers of boards with similar architectures (such as IIT and Weitek).

4.9.3.3 Choosing a Video Adapter for OS/2

The best choice of a video adapter for OS/2 is an important consideration. Points to keep in mind include:

- If you are purchasing a video solution for a large number of systems, always standardize on one video board model for all systems. This will save a great deal of trouble now and in the future.
- Non-accelerated boards (such as Tseng ET4000, Trident 8900/9000, Headlead HT209, Cirrus 5424 and lower, and WD 90C30) are much easier to program than accelerated boards (such as S3, XGA, IIT AGX, Cirrus 5426, ATI Mach and WD 90C31).

  Therefore, display drivers for operating systems like OS/2 will be available more quickly for these boards. Also, DOS applications rarely use the accelerated functions of the newer accelerated boards, so they may actually be slower for DOS applications.
- When using a Graphical User Interface like OS/2, however, accelerators take a large burden off the CPU. CPU time that would have been spent moving pixels around on the screen can now be used for applications.

  In a multitaking operating system like OS/2 where the CPU can be working on many things at once, this is very advantageous.

When picking a particular implementation, it is always best to go with a board that is likely to sell large quantities, has outstanding software driver support from the manufacturer, and which does not use unique or proprietary extentions onto the base generic chipset.

4.9.4 Understanding Video Monitors

Computer monitors have different capabilities. VGA monitors can only display 640x480 pixels with 60Hz vertical refresh. SVGA monitors range from displaying 800x600 at 56Hz up to 1280x1024 at 72Hz and beyond.

Here are three typical monitor profiles:

VGA monitor
640x480	60Hz vertical refresh non-interlaced
800x600	not capable
1024x768	not capable

Low-end SVGA
640x480	72Hz vertical refresh non-interlaced
800x600	56Hz vertical refresh non-interlaced
1024x768	45Hz vertical refresh interlaced

High-end SVGA
640x480	72Hz vertical refresh non-interlaced
800x600	72Hz vertical refresh non-interlaced
1024x768	72Hz vertical refresh non-interlaced

Interlaced monitors and lower refresh rates can cause annoying flicker on the screen, which is can also be damaging to the eyes. So users will often spend

more in order to purchace a high-end monitor which does not flicker. Each different monitor model may vary in terms of the refresh rates it supports.

The difficulty arises because there is no standard way for the SVGA video board to detect what kind of monitor is attached to the system. Typically, a video board will assume that the monitor is a low-end SVGA monitor. So, if the user runs a VGA monitor with the board, and attempts to run a high resolution mode, the monitor will go out of sync. If the person has a SVGA monitor, it will be driven at 45Hz interlaced at 1024x768, regardless of what the monitor is capable of.

Because these SVGA boards were designed for DOS and Microsoft Windows, the way they handle this problem is to supply a DOS utility which the user runs and uses to select the type of monitor attached to the system. Some also have MS Windows drivers hardcoded by refresh rate. This utility then modifies the state of the video board to reflect the type of monitor.

These utilities can be very confusing, and it is common to have to run experiment by trial and error between running the utility and testing in graphics mode, only to find the monitor out of sync and having to reboot and run the utility again to try another setting. Once the utility is configured properly, it is placed in the AUTOEXEC.BAT file or in the CONFIG.SYS file, so it is executed every time the computer is booted. This utility varies from board to board (not just chipset to chipset), as each individual SVGA board may have to be programmed differently to enable it to take full advantage of a particular monitor type.

4.9.4.1 Video Monitors and OS/2

The approach of OS/2 is to use the SVGADATA.PMI file architecture. To properly configure the monitor, the DOS based utilities are used which access the video adapters's BIOS to put the board in the proper state, then read in the state, store it, and use it for mode sets in protect mode under OS/2.

The way this is accomplished is using the DOS program SVGA.EXE (by running SVGA ON), which reads the current state of the video adapter, and places it in a ASCII file. The format of this file is based upon the VESA protect mode SVGA standard. The file can be edited by a programmer or another program, and is interperited at boot time and stored in base video data structures. This SVGA.EXE utility is executed and proper data generated before base video will provide any SVGA services. Because the program attempts to capture the exact state of the adapter at the time it is run, if the user runs his DOS monitor configuration utility before running SVGA.EXE, OS/2 captures the state of the board when it is properly configured for the monitor.

Thus, OS/2 can use the fact that these video boards all come with DOS support to assist OS/2 in dealing with the boards and monitors properly.

The data file ("Protect Mode Interface" file, or .PMI) is editable, so manufacturers of frame buffer SVGA boards can potentially add OS/2 support for their board simply by editing the data file.

4.9.5 Understanding Display Driver Configurations

There are typically five different display drivers which need to be provided for a specific display configuration

- A base video handler for full-screen OS/2 sessions
- A PM display driver for OS/2 PM applications
- A DOS virtual display driver
- A full-screen WIN-OS/2 display driver
- A seamless WIN-OS/2 display driver

Each display driver may be implemented as a combination of DLLs and device drivers.

In addition, the appropriate set of fonts for the display is needed.

The statement SET VIDEO_DEVICES=VIO_xxx,VIO_yyy in the \CONFIG.SYS file is used to identify the primary and secondary displays used by OS/2. There should be corresponding lines of the form SET VIO_xxx=(Drivers,DLLs) in the \CONFIG.SYS for each entry in the SET VIDEO_DEVICES statement; these specify the device drivers and DLLs to be loaded for each display.

Some information is also held in the OS2.INI file, and is thus not readily viewable by the end user (however, the Prf* APIs can be used from within a program).

Table 18. Types of Video Display Drivers

Driver Type	Purpose	Example files for VGA	Specified in
Base Video Handler	Full-screen OS/2 sessions	BVHVGA.DLL BVHWNDW.DLL	SET VIO_*GA= statement in \CONFIG.SYS
Presentation Driver	PM sessions	IBMVGA32.DLL IBMDEV32.DLL	OS2.INI file. Some renaming of files during installation (for example to IBMDEV32.DLL) also occurs. Ring 0 drivers can be specified in the \CONFIG.SYS file using BASEDEV= and DEVICE= statements
DOS virtual display driver	Virtual DOS sessions	VVGA.SYS	DEVICE= statement in \CONFIG.SYS
WIN-OS/2 display driver	Full-screen WIN-OS/2 sessions	VGA.DRV	DISPLAY.DRV= statement in \OS2\MDOS\WINOS2\SYSTEM.INI
WIN-OS/2 seamless display driver	Seamless WIN-OS/2 sessions	SVGA.DRV	SDISPLAY.DRV= statement in \OS2\MDOS\WINOS2\SYSTEM.INI

Information on how to install the display drivers is stored in the *.DSC files in the \OS2\INSTALL directory, and also in the *.DSP files on the OS/2 display driver diskettes (or in the DISP_x subdirectories on the CD-ROM). This information is used by the system to copy across the correct drivers to the fixed disk, and to modify the OS/2 and WIN-OS/2 initialization files. The *.DSP files are not copied across to the fixed disk at installation time.

4.10 Pentium Exploitation

Intel's new Pentium chip, the successor to the 386 and 486 microprocessors, provides a range of high-performance features. These include:

- 64-bit data path
- Dual instruction pipelines
- Enhanced floating point unit
- Separate 8KB data and instruction write-back caches
- Enhanced virtual 8086 mode (virtual mode extensions)

The Pentium is upwardly compatible with the 386 and 486 microprocessors, and so OS/2 2.1 is able to use the intrinsic higher performance of the Pentium without any special recompilation or modifications.

In addition, changes have been made to OS/2 2.1 in order to exploit the Pentium chip. The first phase of this exploitation improves the performance of DOS sessions through the use of virtual mode extensions. Pentium now virtualizes system interrupt flags, improving its support of virtual 8086 mode applications (such as DOS applications) by eliminating the need for exception handling routines.

4.11 Preloaded IBM Hardware Systems

OS/2 2.1 is preloaded onto a number of IBM and PCM systems.

At the time of publication (May 1993), the following IBM PS hardware systems were preloaded with OS/2 2.1

Table 19. IBM PS Hardware Systems Preloaded with OS/2 2.1

Machine Type	Model	Preloaded OS/2 2.00.1	Preloaded OS/2 2.1
PS/2 Model 9556	0B6, 0BA	1	Y
PS/2 Model 9556	0EG, 0EA		Y
PS/2 Model 9557	0B6, 0BA	1	Y
PS/2 Model 9557	0EG, 0EA		Y
PS/2 Model 9576	0UG, 0UA, 0NG, 0NA	1	Y
PS/2 Model 9577	0UG, 0UA, 0UF, 0NA, 0NF	1	Y
PS/2 Model 9585			Y
PS/2 Model 9590	0LA	1	Y
PS/2 Model 9590	0LG		Y
PS/2 Server 195, 295			Y
PS/VP 2 Model 6382	F51, K51, M51	2	3
PS/VP 2 Model 6384	F51, K71, M71, W71	2	3
PS/VP 2 Model 6387	M71, W91	2	3
ThinkPad 700	700, 700C	1	Y
ThinkPad 720	720, 720C		Y

Table Notes

1. Preloaded OS/2 2.00.1 now replaced by preloaded OS/2 2.1
2. Preloaded OS/2 2.00.1 interim preload until S3 SVGA display driver available
3. Preloaded OS/2 2.1 once S3 SVGA display driver becomes available

The PS/ValuePoint 2 systems have an interim preload of OS/2 2.00.1. This will change to OS/2 2.1 once the S3 SVGA display driver becomes available. In addition, PS/VP2 systems preloaded with OS/2 2.00.1 contain a free upgrade coupon for OS/2 2.1 and the S3 display driver.

PS/ValuePoint 1 systems are not preloaded with OS/2 2.1, but only with OS/2 2.00.1.

Chapter 5. Control Program Enhancements

Some enhancements have been made to the Control Program in OS/2 2.1. These include:

- Enhancements for local systems management, including the ability to log trap screens to the hard file and initiate process-level dumps
- Support for applications which demand a specific OS/2 2.0 version number
- Format utility support for P-ROM optical disks
- XCOPY enhancements to copy hidden files and empty directories

5.1 Systems Management Enhancements

Two systems management enhancements have been included in OS/2 2.1:

- The ability to log trap screens to the fixed disk.

  This is useful in environments where there is no local operator.

- Process-level dumps can be initiated.

  This avoids dumping the entire OS/2 memory contents if it is known which process a dump is needed for.

Documentation on these features was not available at the time of publication, but will be provided in the future through the OEMI documentation from IBM.

5.2 OS2VER File

Many 32-bit OS/2 applications specifically query the OS/2 version number, in order to check that they will be able to run. Some of these applications check that the OS/2 version number is 2.0, rather than 2.0 or higher. These applications may refuse to run on OS/2 2.1 because the version number is different, rather than for any other reason.

In order to enable these applications to run on OS/2 2.1 without any application upgrades, a capability has been added to OS/2 2.1 to enable these applications to think they are running on OS/2 2.0.

This has been achieved by:

- Specifying a list of applications which need the specific OS/2 2.0 version number
- Altering the APIs which return a version number so that these applications receive OS/2 2.0 rather than OS/2 2.1 as the reply.

A new file, OS2VER, has been added to the root directory of the boot drive, and it contains a list of programs which are to be lied to about the version number, and the version number to reply with.

An initial list of these is provided in a default OS2VER file in the root directory of the boot drive. This file has hidden, system and read-only attributes set.

To add more entries to the OS2VER file, follow these instructions:

1. At an OS/2 command prompt for the root directory of the boot drive, type:
   ATTRIB -h -r -s OS2VER

2. Use an editor to add lines to the file. The typical format is:
   20=PROGRAM.EXE
3. Save and exit the file.
4. Type **ATTRIB +h +r +s OS2VER**.

DosGetVersion and DosQuerySysInfo will return OS/2 2.0 rather than OS/2 2.1 if any of these programs ask for the version number.

5.3 Format Utility Support for P-ROM Optical Disks

As described in section 4.4.4, "3.5" Enhanced Rewritable Optical Drive Support" on page 46, the new 3.5" Enhanced Rewritable Optical Drive includes support for the P-ROM (partial read-only memory) optical disks as well as the MO and O-ROM optical disks previously supported.

The format utility has been enhanced to format P-ROM optical disks in this drive. P-ROM optical disks are formatted at 122MB capacity. This is in addition to the current capability to format MO optical disks at 127MB capacity.

The format utility is typically started from the pop-up menu for the Optical Drive icon in the Workplace Shell.

5.4 XCOPY Enhancements

The XCOPY utility provided the capability to copy whole sets of files, including directories and subdirectories. However, XCOPY in OS/2 2.0 did not copy system or hidden files, nor did it retain the read-only attribute on copying. In addition, it was not possible to overlay system, hidden or read-only files with new versions using the XCOPY command.

This limited the use of XCOPY for copying and replicating parts of OS/2 2.0, such as the desktop, and the hidden system OS/2 files.

XCOPY has been enhanced in OS/2 2.1 to include these features. In order to use any of these new features, extra parameters must be specified. This maintains compatibility of operation with the OS/2 2.0 XCOPY whilst enabling the new features to be exploited.

XCOPY will now:

- Copy hidden files (if the /H parameter is specified).
- Copy system files (if the /T parameter is specified).
- Retain the read-only attribute on files copied (if the /R parameter is specified).
- Overlay system, hidden and read-only files with new copies (if the /O parameter is specified),

Chapter 6. MVDM Enhancements

In the early days of OS/2, initial support was provided to enable DOS applications to run under OS/2. A single DOS session, known as the DOS Compatibility Box, was developed in which there was a fixed amount of memory available to DOS applications.

In OS/2 2.0, a significant milestone was achieved by exploiting the Virtual 8086 mode of the Intel 386 processor family to create the Multiple Virtual DOS Machine (MVDM) component. This enabled multiple DOS applications to execute concurrently, each in its own virtual DOS machine where it was protected from all the other applications and from the operating system. OS/2 2.0 also provided the ability to tailor the DOS environment for each application, and to manage DOS applications from the Workplace Shell.

OS/2 2.1 includes further enhancements to enable a more tailored DOS environment to be created for advanced DOS applications. Multimedia DOS applications can use the dual-thread feature to run smoothly, and have updated MSCDEX CD-ROM compatibility available to them. Tailored AUTOEXEC.BAT files can be set up for specific applications, and the DPMI support has been upgraded.

This chapter describes these new features in detail. For a good understanding of the DOS environment in OS/2, please refer to *OS/2 Version 2.0 - Volume 1: Control Program* and *OS/2 Version 2.0 - Volume 2: DOS and Windows Environment*.

6.1 Dual-Thread Support in MVDMs

DOS multimedia applications can be especially demanding of the DOS environment they are running in, since they need to ensure smooth display of pictures and sound reproduction while also reading large amounts of data from disk or CD.

The MVDM support in OS/2 2.0 was single-threaded, and applications that were dependent on software interrupts (such as timer ticks) for updating the screen could be suspended while waiting for a file system data access to complete. The visible effect of this was that the display or sound output of these applications would not appear smooth.

OS/2 2.1 enhances the MVDM support by enabling a second thread to be started. This dual-thread capability enables timer tick interrupts to be received by the application while it is also waiting for a file system request to complete.

This means that DOS multimedia applications, such as *Sherlock Holmes Consulting Detective***, will now run smoothly in a VDM under OS/2 2.1.

6.1.1 Setting Up DOS Applications for Dual-Thread

To set up a DOS application to use dual-thread, follow these steps:

1. Display the context menu of the DOS application object by clicking the right mouse button on the application icon.
2. Display the **Open** options by clicking the left mouse button on the right pointing arrow within the **Open** box as shown in Figure 38.

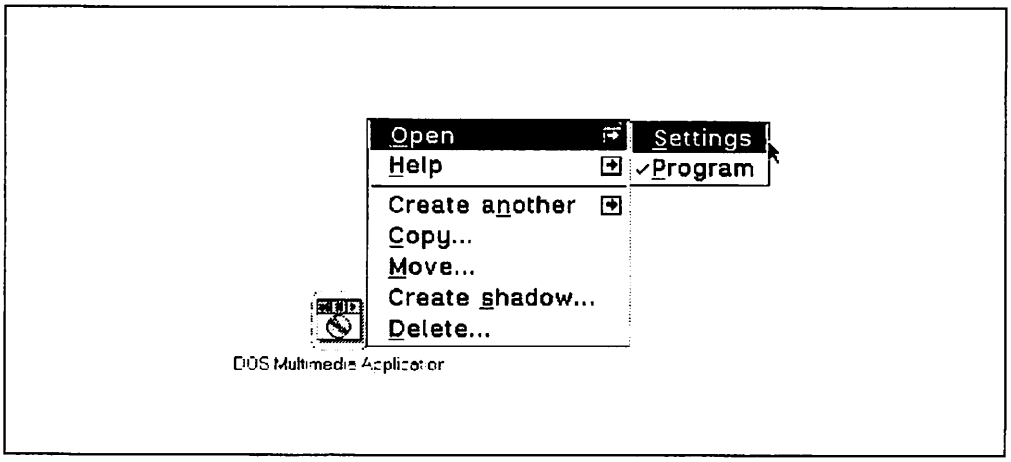


Figure 38. Setting up MVDM Dual-Thread: Pop-Up Menu

3. Click the left mouse button on the **Settings** option and a **Settings** notebook for the DOS application object will be displayed.
4. Click the left mouse button on the **Session** tab to jump to that page as shown in Figure 39.

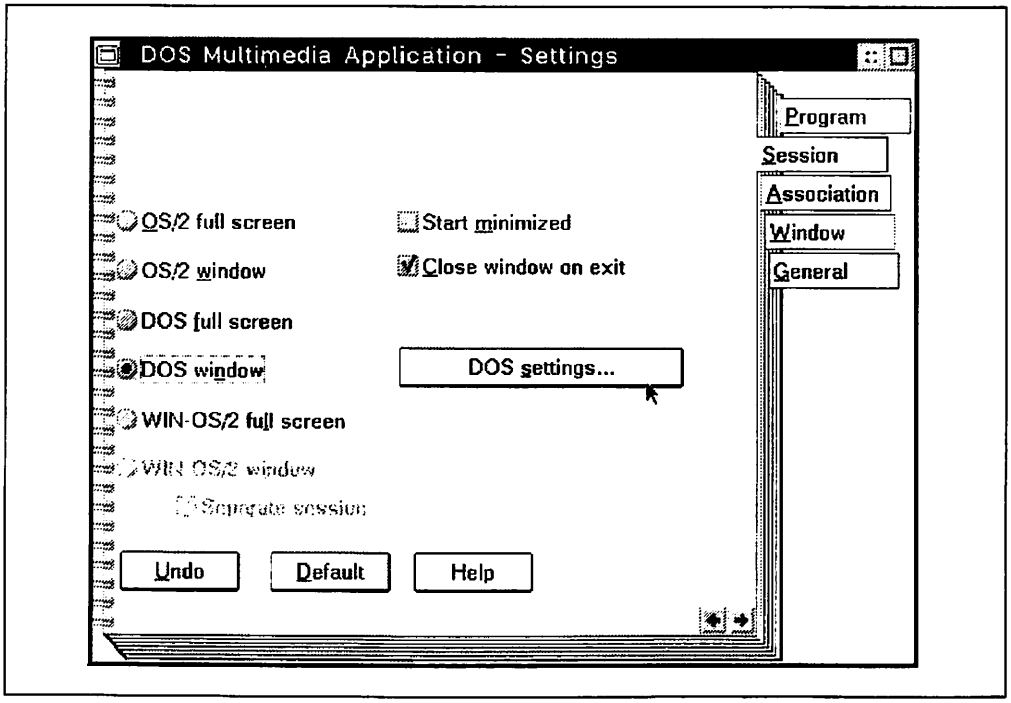


Figure 39. Setting Up MVDM Dual-Thread: Session Page of Settings Notebook

5. Click the left mouse button on the **DOS** settings box and the DOS Settings panel is displayed.

6. Scroll down the **DOS Settings** list box (by clicking the left mouse button on the down pointing arrow in the scroll box) until the **INT_DURING_IO** definition appears.
7. Click the left mouse button on the **INT_DURING_IO** item to set the dual-thread function and the selected definition will be highlighted. The default value is **Off** and a brief description of the definition is displayed as shown in Figure 40.

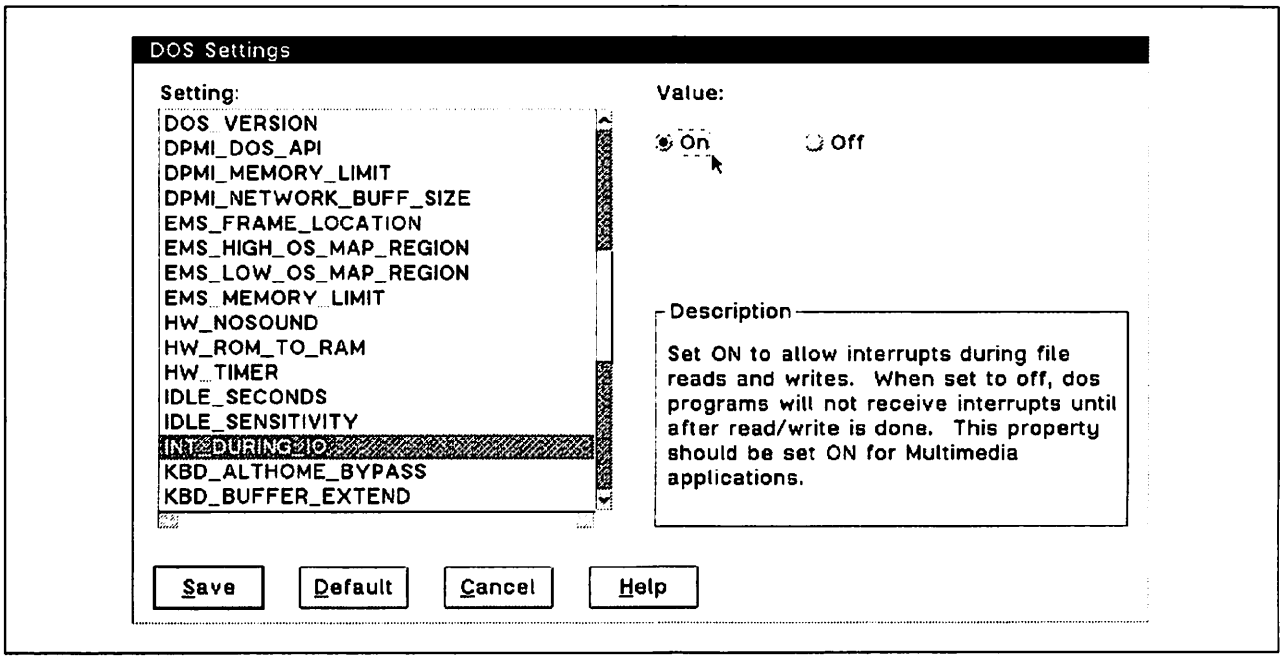


Figure 40. Setting Up MVDM Dual-Thread: INT_DURING_IO Setting

8. Click the left mouse button on the **On** radio button to enable dual-thread operation.
9. Click the left mouse button on the **Save** box to register the settings made and close the notebook by double-clicking on the system menu.
10. The DOS program may now be restarted by double clicking the left mouse button on the object icon and the dual-thread function will be active.

6.1.2 Implementation

DOS multimedia applications are faced with the challenge of how to produce video and audio output fast enough so that it appears smooth to the user, while also reading large amounts of data from disk or CD.

Some of these applications solve this by hooking into the timer tick interrupt, and using this to regularly update the screen and produce audio output, even if the application is currently reading data from disk, as shown in Figure 41.

A good example of such a DOS multimedia application is *Sherlock Holmes Consulting Detective*.

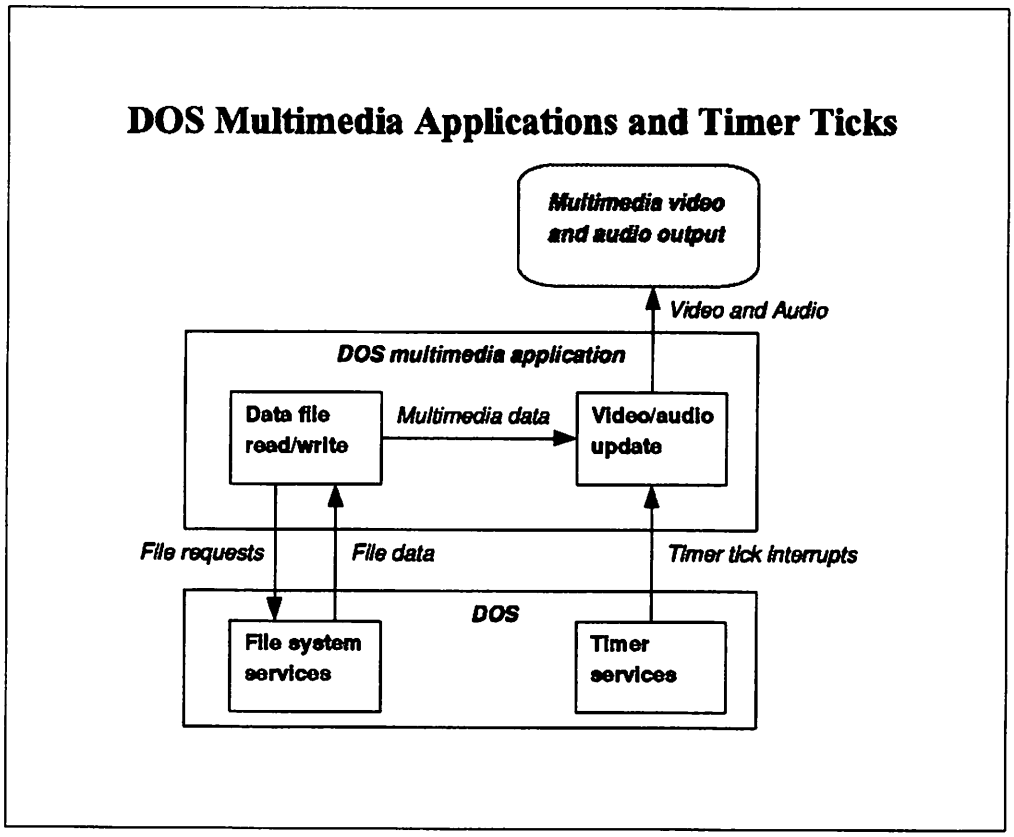


Figure 41. DOS Multimedia Applications and Timer Ticks

This approach works well in the native DOS environment. However, in the virtual DOS environment of an OS/2 2.0 VDM, there was the drawback that the single thread could be blocked while waiting for a file system request to complete, and thus was not interruptible by the emulated timer tick interrupt. This is shown in Figure 42 on page 69.

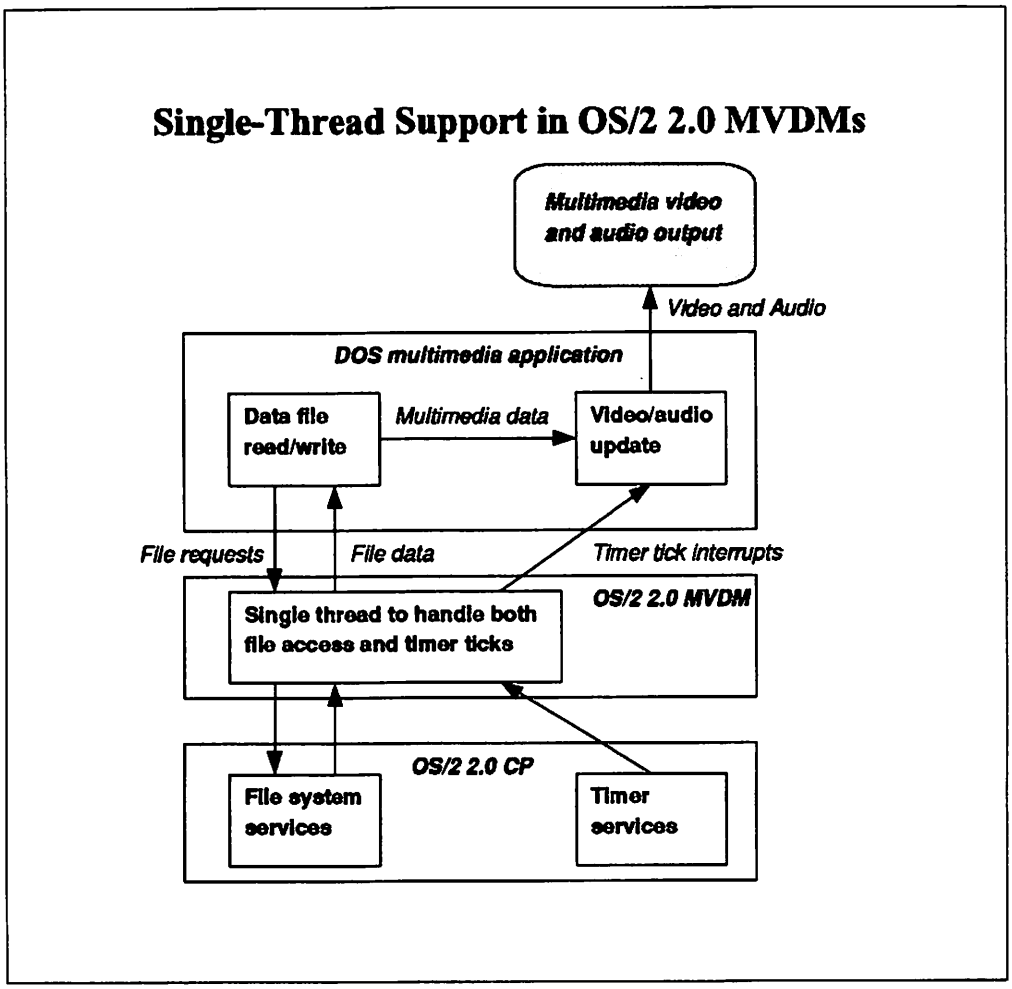


Figure 42. MVDM - Limitations of Single-Thread Support under OS/2 2.0

The solution introduced in OS/2 2.1 is the ability to run a second thread in the VDM. The first thread handles file system requests, allowing the second thread to react to timer tick interrupts and thus regularly produce video and audio output. This is shown in Figure 43 on page 70.

This dual-thread support needs more memory and CPU time than the single-thread MVDM, and so dual-thread INT_DURING_IO should only be selected when needed by the application. However, the extra memory requirement does not reduce the memory available to the DOS application within the VDM.

It should be noted that the dual-thread support is not a general multiple-thread MVDM implementation, but has been written and optimized specifically to enable DOS multimedia applications to run smoothly in a VDM under OS/2 2.1.

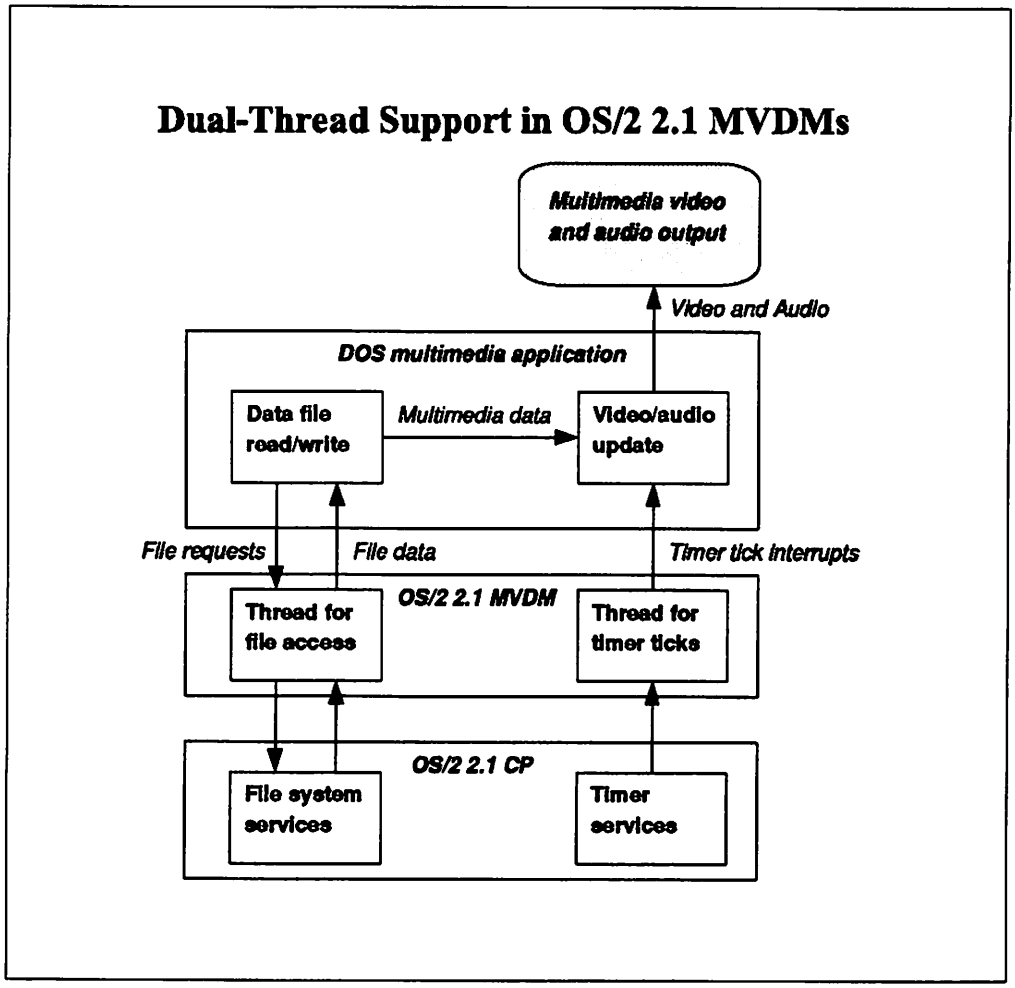


Figure 43. MVDM - Benefits of Dual-Thread Support under OS/2 2.1

6.2 DOS_AUTOEXEC Setting

A new DOS setting has been added in OS/2 2.1. The DOS_AUTOEXEC setting enables a specific DOS command (.BAT) file to be executed when a VDM is created. This feature can be used to tailor the DOS environment and is particularly useful if programs other than device drivers need to be run when creating the VDM.

The DOS_AUTOEXEC setting can be selected from the DOS Settings panel in the normal way. When DOS_AUTOEXEC has been highlighted in the list box, the user can change the name of the DOS command file which is to be run, as shown in Figure 44 on page 71

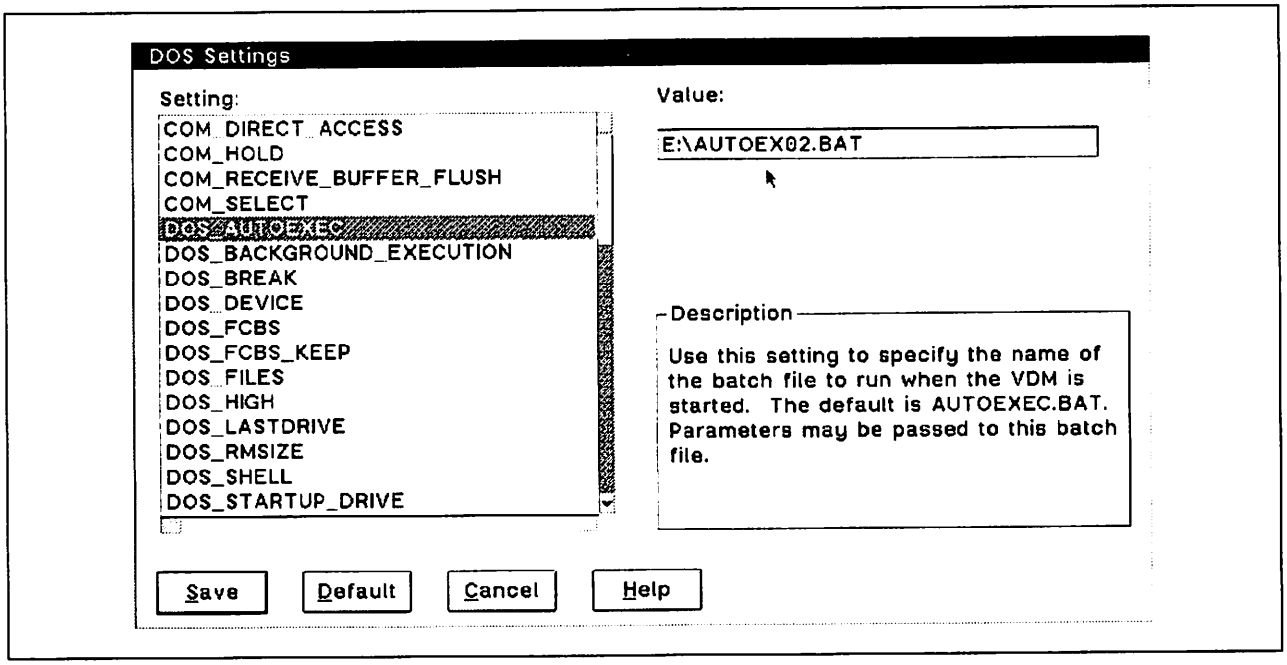


Figure 44. DOS_AUTOEXEC Setting

The default is to run the AUTOEXEC.BAT file in the root directory of the boot drive, and this is also the initial default displayed when changing the DOS_AUTOEXEC setting.

6.3 DPMI 1.0 Subset Support

DPMI is a protected mode programming interface for DOS applications, allowing these applications to run on the 80286, 80386 and 80486 processor in protected mode, while utilizing the real mode services of the operating system and device drivers. It defines a subset of DOS and BIOS calls that can be made by DOS programs running in the protected mode via software interrupt, INT 31h.

In OS/2 2.0, DPMI Version 0.9 was supported, primarily in order to enable WIN-OS/2 3.0 to be implemented.

With the release of OS/2 2.1, the DPMI support has been upgraded to a defined subset of DPMI 1.0.

The DPMI support has been designed with DPMI 1.0 in mind, but only some of the DPMI 1.0 features have so far been implemented, in order to support specific applications.

In particular, the OS/2 2.1 DPMI 1.0 Subset enables support for the following applications:

- WIN-OS/2 3.1, both standard mode and enhanced compatibility mode
- AutoCad** for DOS
- Extended DOS version of PC Support/400 (V2R3), which provides AS/400 terminal emulation sessions but requires multiple DPMI client support.

OS/2 2.0 included multiple DPMI client support to DPMI 0.9 level, but this specification required that the DPMI clients were all either 16-bit or 32-bit, and that they chained interrupts properly. With the OS/2 2.1 DPMI 1.0 Subset, these

restrictions have been removed, which was needed to enable the AS/400 terminal emulation application (provided in PC Support/400 (V2R3)) to run in a MVDM.

The DPMI support has also been enhanced in other ways:

- Some DPMI applications (such as *PORTIA**) which try to directly access the video buffer (using DOS memory segment B800) do not run in an OS/2 2.0 VDM. DOS Emulation has been enhanced in OS/2 2.1 to create an alias selector if a DPMI application tries to directly access the video buffer.
- Some DOS compilers which use DPMI could cause a DPMI "Operating System Error" in an OS/2 2.0 VDM, due to an error checking the limits of 32-bit expand down stacks. The MVDM support in OS/2 2.1 has been enhanced to expand down stacks correctly.

6.4 MSCDEX Enhancement to VCDROM

MSCDEX.EXE is the Microsoft DOS CD-ROM Extensions driver which is used in the native DOS environment. It provides an advanced CD-ROM file system driver, and is needed by a number of DOS and Windows applications.

The VCDROM.SYS virtual device driver provides CD-ROM services to DOS applications running in an OS/2 2.0 VDM. Some of the functions of MSCDEX were included in the VCDROM virtual device driver in OS/2 2.0, but this was not a full implementation of the data and audio components.

VCDROM has been enhanced in OS/2 2.1 to include all the data (CD-ROM) and CD audio functionality of MSCDEX, thus enabling DOS and Windows multimedia applications that use these parts of the MSCDEX interface to run in a VDM under OS/2 2.1.

VCDROM also provides CD-XA support for DOS applications, via the CDFS.IFS file system in OS/2 2.1. This CD-XA support is separately implemented to the MSCDEX support in VCDROM.

Chapter 7. WIN-OS/2 3.1 Support and Enhancements

OS/2 2.1 combines the ability to run Windows 3.1 applications with the advanced operating system benefits of OS/2 Version 2. It achieves this by providing an enhanced WIN-OS/2 3.1 environment.

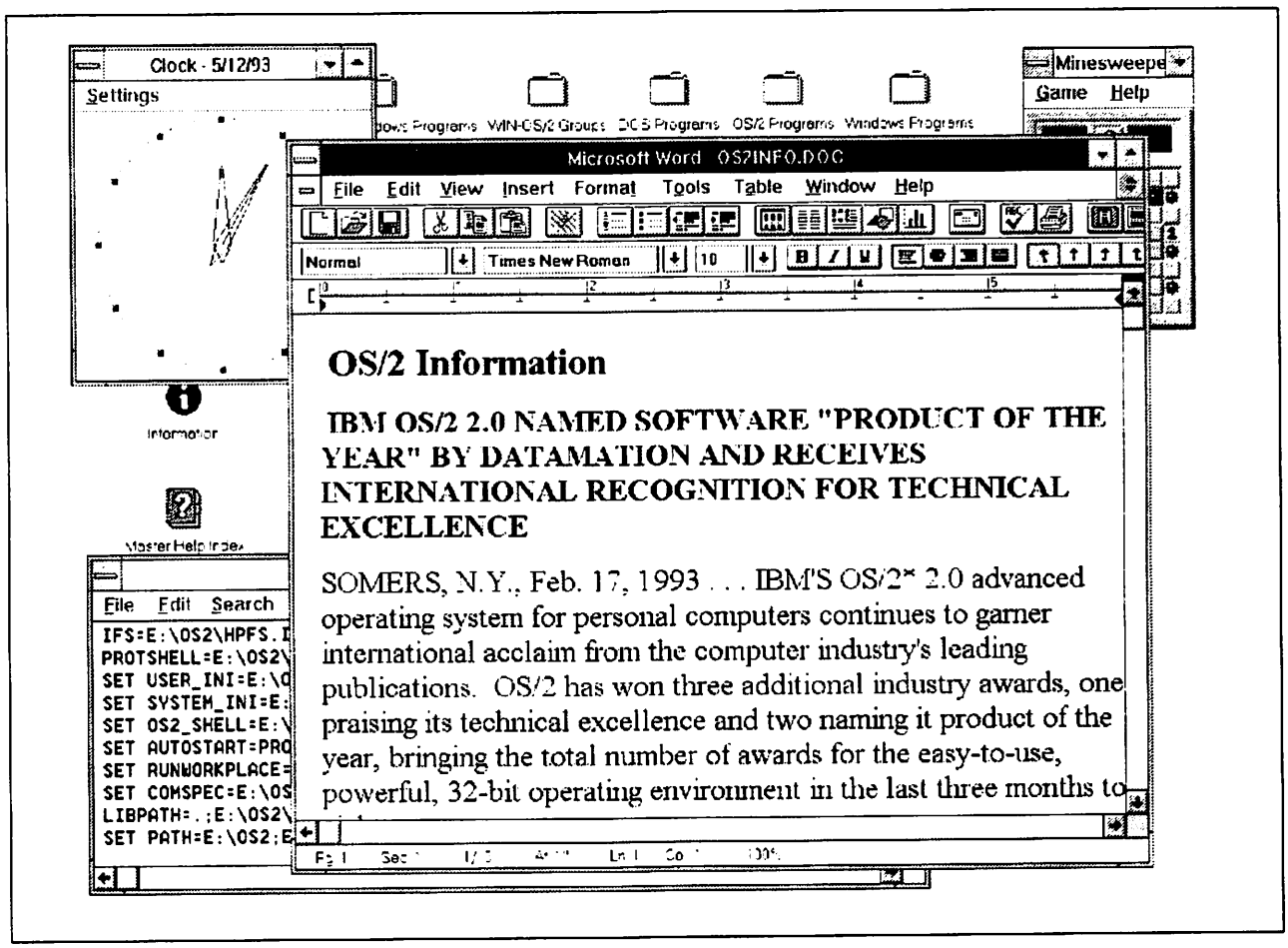


Figure 45. WIN-OS/2 3.1 Running under OS/2 2.1 in XGA Mode

The WIN-OS/2 3.0 support provided in OS/2 2.0 was a major leap forward, and enabled a large number of Windows applications to be run alongside OS/2 and DOS applications on the Workplace Shell desktop.

The aim of the improved WIN-OS/2 3.1 support in OS/2 2.1 is to provide an advanced environment in which to run Windows 3.1 applications:

- Most Windows applications will run, including those that exploit Windows 3.1 features such as OLE (Object Linking and Embedding).
- Performance is comparable to the Windows 3.1 environment.
- Windows applications fit seamlessly alongside DOS and OS/2 applications on the Workplace Shell desktop.
- Clipboard and DDE data exchange facilities are available between Windows and OS/2 applications, and have good performance and ease-of-use.

- The extra benefits of OS/2 Version 2 (such as multitasking and protection between applications) are available to Windows applications running under OS/2 2.1.

The WIN-OS/2 3.1 support in OS/2 2.1 replaces the WIN-OS/2 3.0 support in OS/2 2.0. Although real mode Windows applications are no longer supported under either Windows 3.1 or WIN-OS/2 3.1, most Windows applications have now been updated for Windows 3.1.

This chapter describes the enhancements provided by the WIN-OS/2 3.1 support in OS/2 2.1. For more details of the WIN-OS/2 support common to both OS/2 2.0 and OS/2 2.1, please refer to *OS/2 Version 2.0 - Volume 2: DOS and Windows Environment*.

7.1 Summary of Enhancements

WIN-OS/2 3.1 meets the above objectives, by merging the Windows 3.1 code enhancements with the modified WIN-OS/2 3.0 code, by adding extra seamless display drivers and printer drivers, and by tuning and redesigning various components of WIN-OS/2 in order to increase performance.

WIN-OS/2 3.1 includes the following enhancements:

- Windows 3.1 standard mode environment
- WIN-OS/2 3.1 performance improvements (see 13.5.4, "WIN-OS/2 3.1" on page 202 for more details)
- Enhanced compatibility mode environment, that enables major Windows enhanced-mode applications (such as Mathematica for Windows and Omnipage Professional) to run
- Object Linking and Embedding support
- Windows audio multimedia support
- Addition of the Level 2.5 Adobe Type Manager (ATM) fonts providing an improved level of function
- TrueType font support
- Ability to start DOS and OS/2 applications from within a Windows application
- Seamless WIN-OS/2 3.1 display drivers for VGA, SVGA and XGA
- Support for Windows 3.1 printer drivers
- Redesigned DDE and Clipboard support with increased performance
- Improved WIN-OS/2 setup and configuration
- Inclusion of the File Manager and other Windows 3.1 applets

If OS/2 2.0 is upgraded to OS/2 2.1, WIN-OS/2 3.0 will be automatically replaced by WIN-OS/2 3.1. If WIN-OS/2 3.1 is not selected for installation, then WIN-OS/2 3.0 will be removed.

7.2 WIN-OS/2 3.1 Appearance

When WIN-OS/2 3.1 is installed on an OS/2 2.1 system, a new **WIN-OS/2 Groups** folder can be seen on the desktop, and the **Windows Programs** and **Additional Windows Programs** folders (if present) now use the standard folder icon.

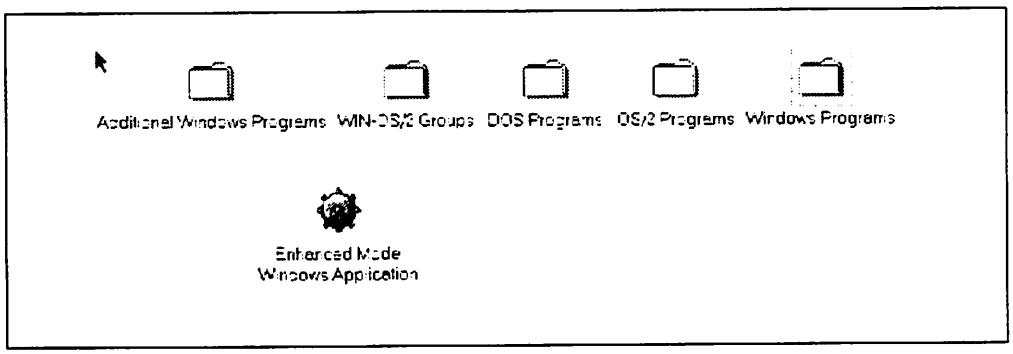


Figure 46. WIN-OS/2 3.1 Folders on the OS/2 2.1 Desktop

The **WIN-OS/2 Groups** folder contains two other folders:

- **WIN-OS/2 Main** contains the Windows utility and setup programs, such as the File Manager and the Control Panel.
- **WIN-OS/2 Accessories** contains the Windows applets, such as Write and Paint Brush.

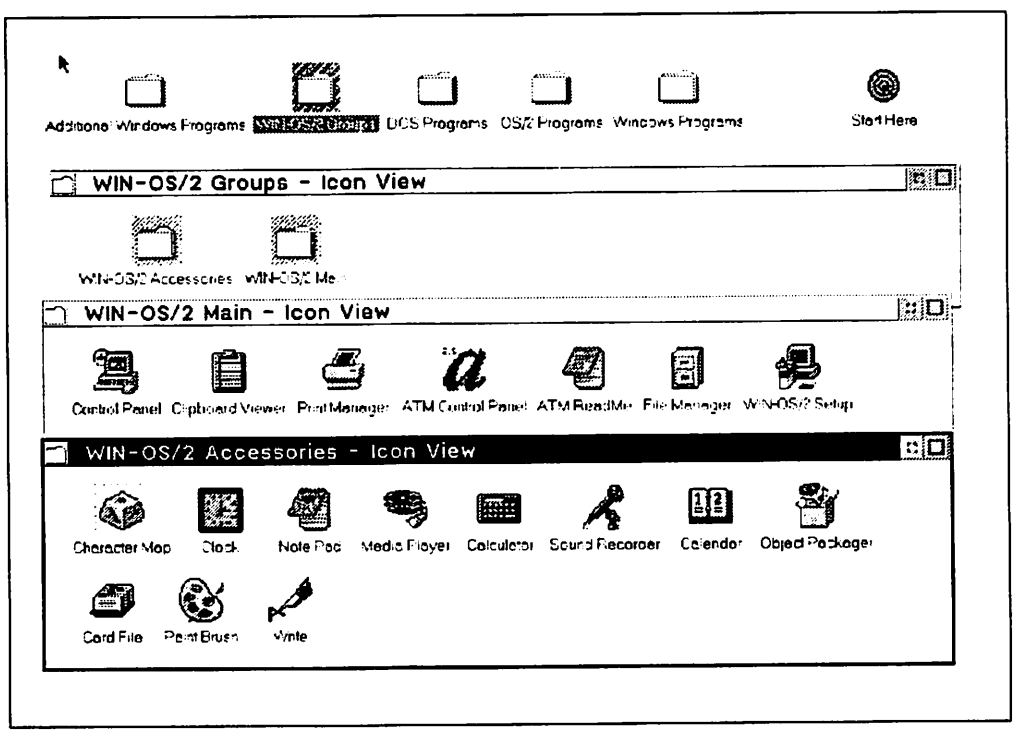


Figure 47. Contents of the WIN-OS/2 Groups Folders

The **Windows Programs** folder contains the WIN-OS/2 3.1 program manager (PROGMAN.EXE), along with any Windows applications defined in the migration database and migrated using the OS/2 2.1 Migrate Applications utility.

The **Windows Additional Programs** folder contains any WIN-OS/2 3.1 applications that are not defined in the migration database but have been specified using the **Add Programs** feature of the Migrate Applications Utility, and then migrated.

Figure 48 shows example contents of these folders.

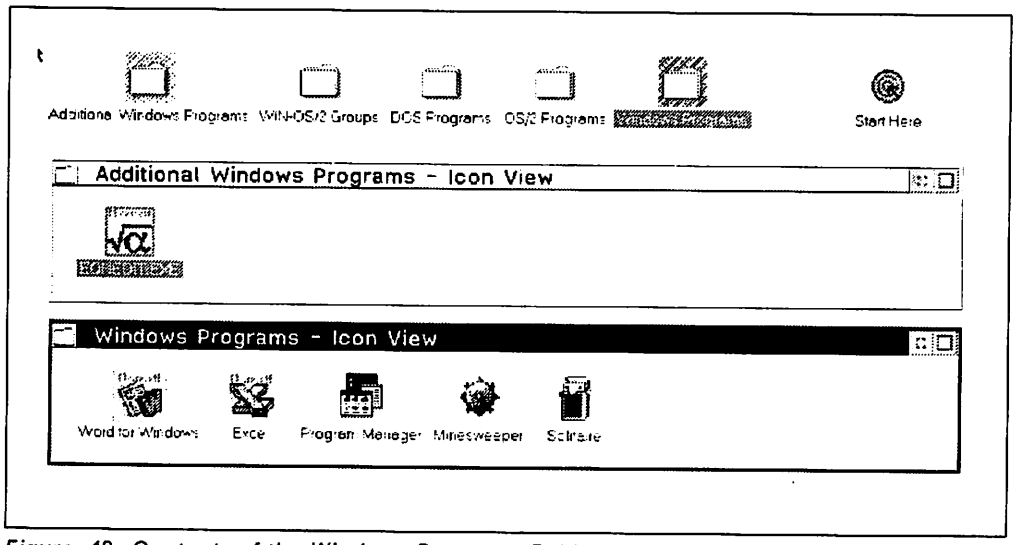


Figure 48. Contents of the Windows Programs Folders

In addition, a WIN-OS/2 Setup object is installed in the System Setup folder (within the OS/2 System folder). This Setup object is used to configure application defaults for Windows applications running under WIN-OS/2 3.1, such as Clipboard and DDE usage.

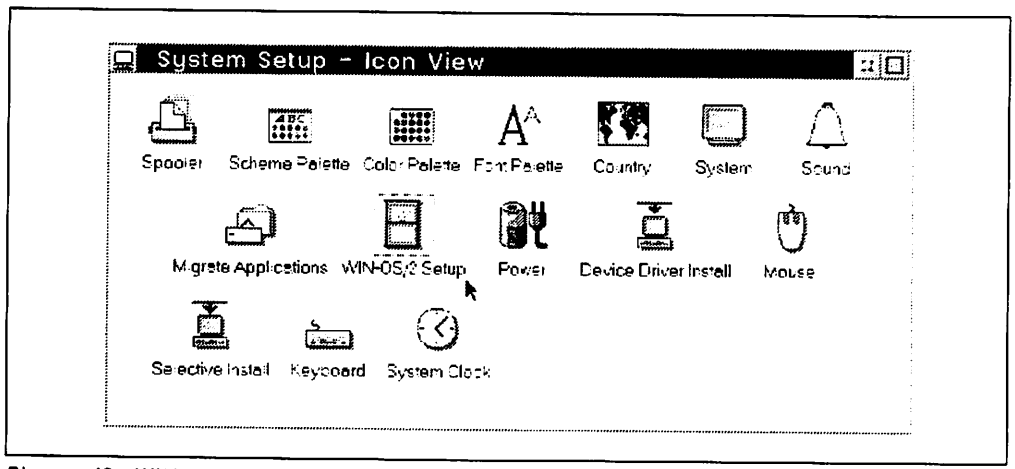


Figure 49. WIN-OS/2 Setup Object in the OS/2 System Setup Folder

Try not to confuse this WIN-OS/2 Setup object with the WIN-OS/2 Setup Windows application in the WIN-OS/2 Main folder, which is used to configure the network options and to add applications to the program groups used by the Windows program manager.

7.3 WIN-OS/2 3.1 Standard Mode

A major enhancement in OS/2 2.1 is the introduction of WIN-OS/2 3.1 standard mode. This enhancement provides the equivalent of a Windows 3.1 standard mode environment and enables a wide variety of Windows applications to run alongside DOS and OS/2 applications, for example, Microsoft Word 2.0 and Microsoft Excel 4.0 as shown in Figure 46 on page 75.

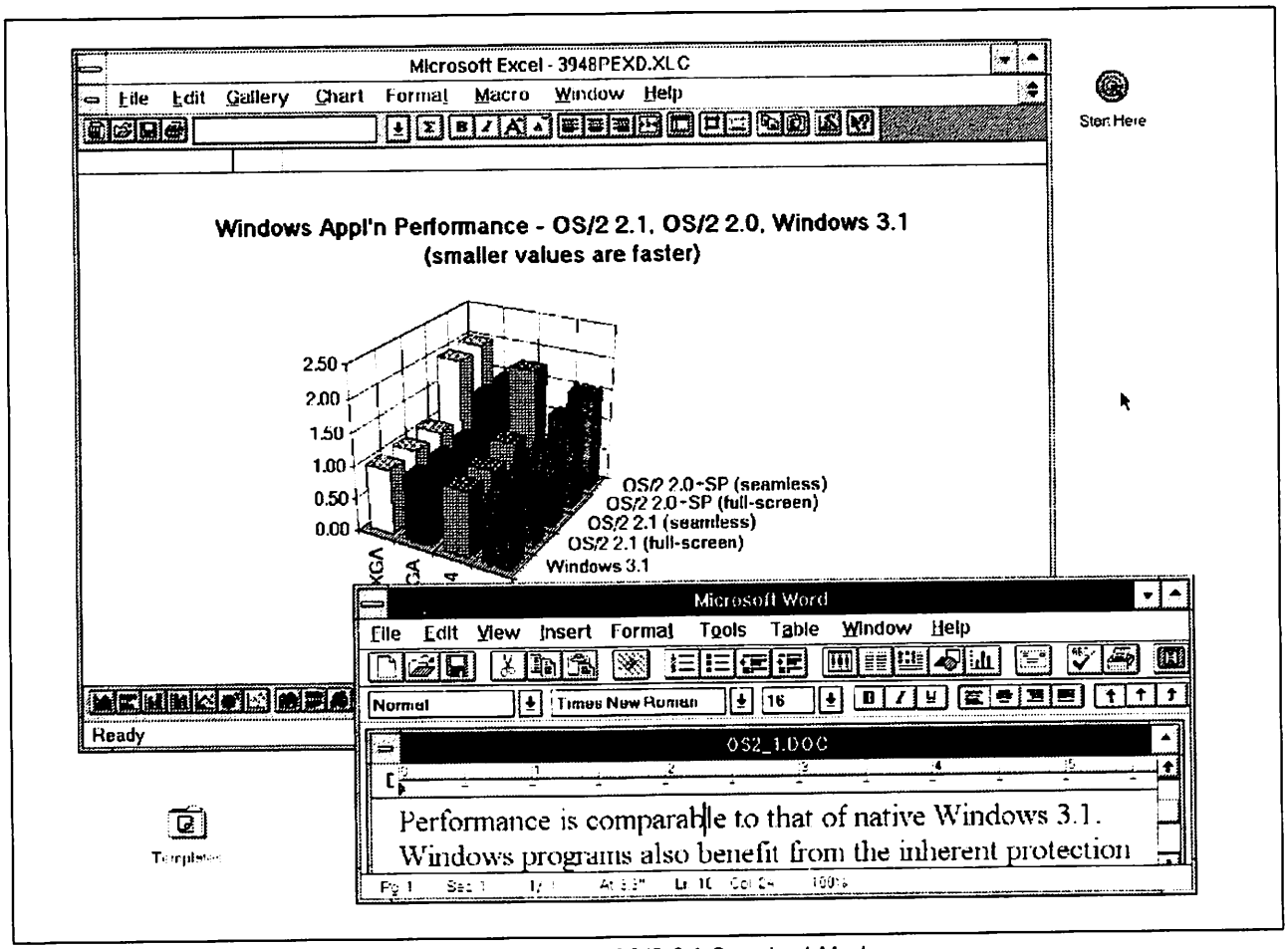


Figure 50. Windows 3.1 Applications Running in WIN-OS/2 3.1 Standard Mode

WIN-OS/2 3.1 standard mode replaces the standard mode of WIN-OS/2 3.0; real mode is no longer supported.

7.3.1 Setting Up Applications for Standard Mode

Windows applications will default to standard mode, and no special procedure normally needs to be followed.

To check that standard mode has been selected, display the WIN-RUNMODE settings option from within the WIN-OS/2 Settings section of the Settings notebook.

The WIN_RUNMODE settings option has changed from WIN-OS/2 3.0, and there are now two radio buttons for selection of either WIN-OS/2 3.1 standard mode or WIN-OS/2 3.1 enhanced compatibility mode, as shown in Figure 51 on page 78. This will normally be set to standard mode, and this is the recommended option unless the application requires 386 enhanced mode.

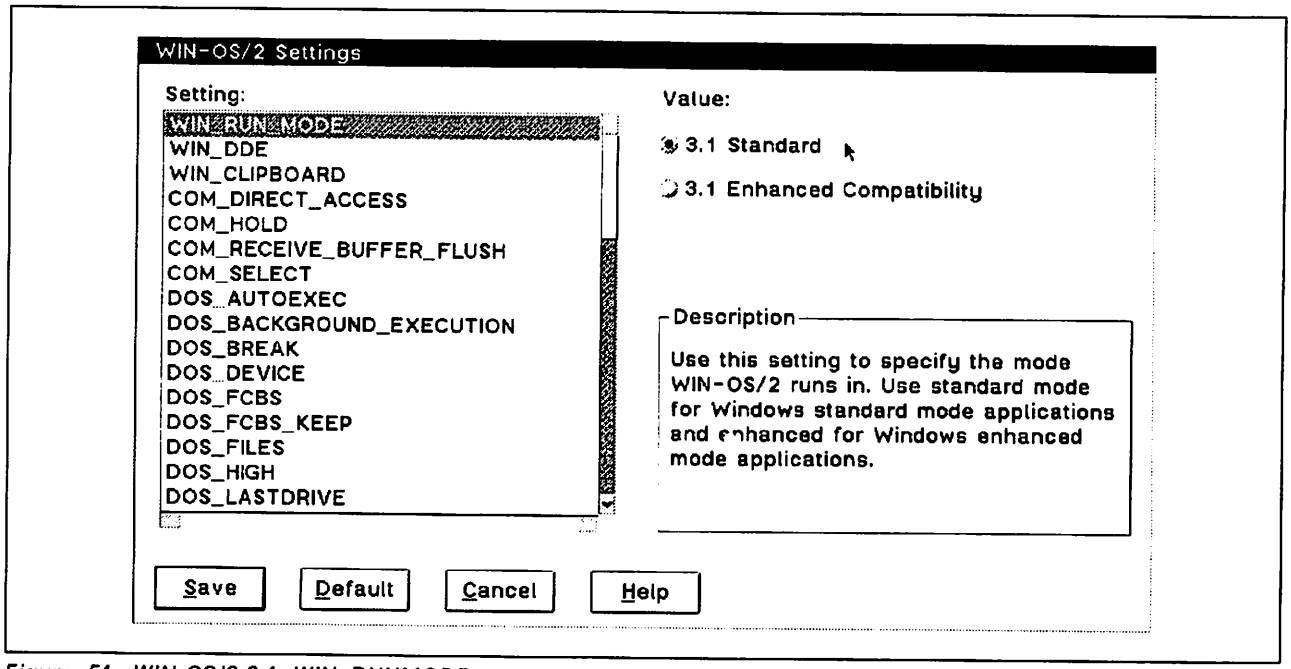


Figure 51. WIN-OS/2 3.1: WIN_RUNMODE

The default AUTOEXEC.BAT file is normally used for running Windows applications in a VDM. However, the DOS_AUTOEXEC setting can be used to execute a specific DOS batch command file when the Windows application is started. Use of this setting is exactly the same as for DOS applications and is described in more detail in 6.2, "DOS_AUTOEXEC Setting" on page 70.

7.3.2 Implementation

WIN-OS/2 3.1 runs a modified copy of the Windows 3.1 code in a VDM. These modifications are necessary because Windows 3.1 can only act as a DPMI host, and thus it cannot run in an OS/2 Version 2 VDM where it would have to be a DPMI client underneath OS/2 (acting as the hypervisor).

Changes are also necessary in order to share the display with Presentation Manager and to integrate into the Workplace Shell desktop.

A simplified overview of the WIN-OS/2 3.1 module structure is shown in Figure 52 on page 79.

In a similar structure to Windows 3.1, the GDI module handles the graphics, the USER module manages the user interface and windowing, and the OS2K386 module performs the extra kernel functions. These interact with device drivers.

There are two different WIN-OS/2 display drivers for each video configuration, one for full-screen and one for seamless operation.

Coordination between WIN-OS/2 3.1 and the rest of OS/2 is handled via the VWIN.SYS virtual device driver. In addition, the seamless mode of operation is coordinated by the WINSHELD module in WIN-OS/2 and the PMSHIELD module in Presentation Manager. In effect, WIN-OS/2 and PM each own part of the display and write directly to the hardware. The job of WINSHELD, PMSHIELD and VWIN is to coordinate these actions and ensure that each component only writes to the part of the display that it owns.

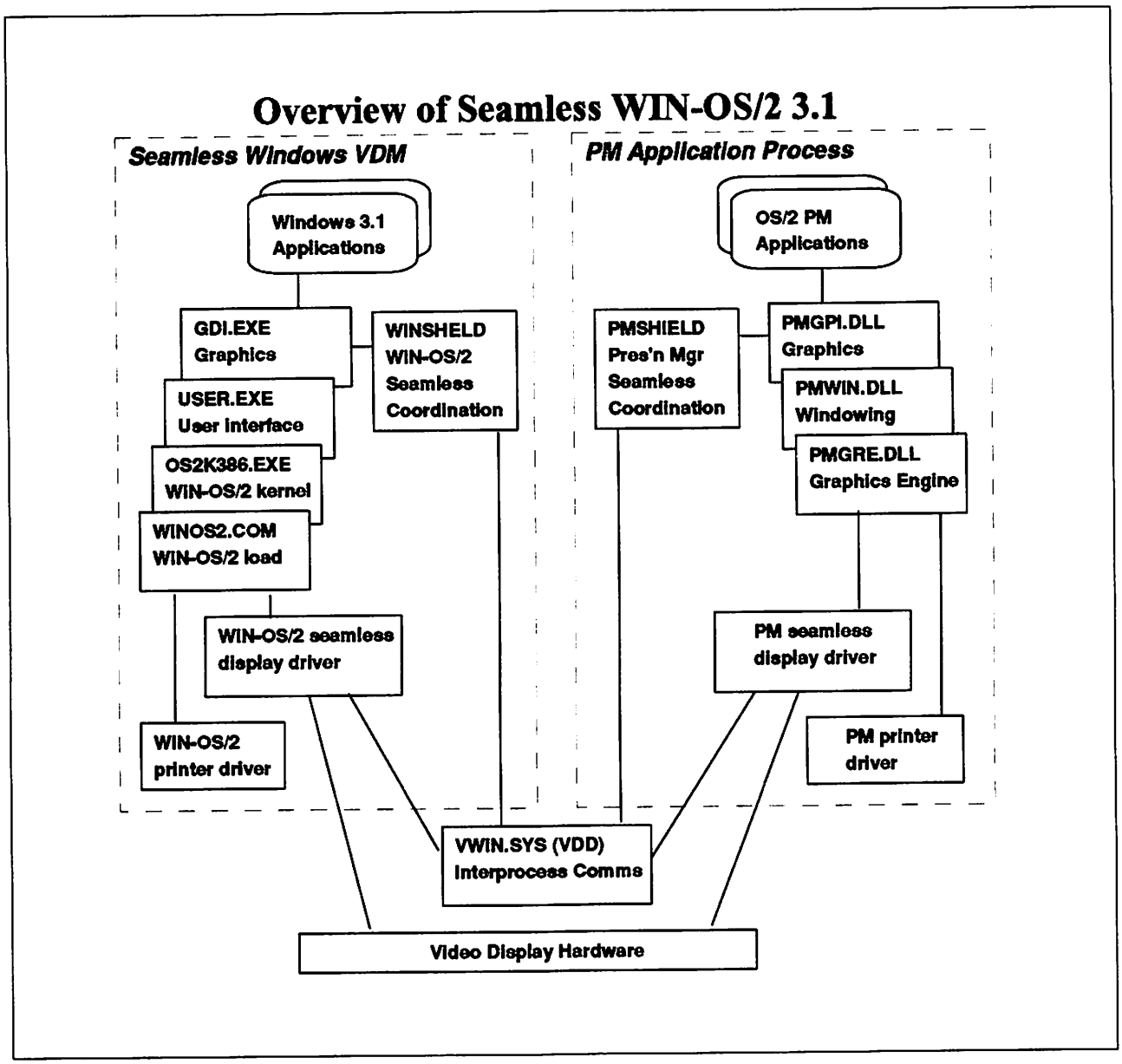


Figure 52. Overview of WIN-OS/2 3.1 Seamless Windows

Some of the key changes to the major modules in WIN-OS/2 3.1 are as follows:

- **WINOS2.COM**

  This module starts WIN-OS/2 3.1 and loads the rest of the WIN-OS/2 3.1 environment. Normally this command is not executed directly, since using a program object enables the VDM settings (such as DPMI memory) to be configured better.

- **OS2K386.EXE**

  This new WIN-OS/2 3.1 kernel includes all the functionality of the Windows 3.1 386 kernel (KRNL386.EXE) and replaces the previous WIN-OS/2 3.0 kernel (OS2K286.EXE). This kernel contains the memory, program loading and scheduling parts of WIN-OS/2 3.1.

  OS2K386.EXE uses the DPMI interface for memory and interrupt management. It executes in full screen and seamless modes, using the current VDM architecture with DOS emulation. A new API, RegisterShield,

has been added to the kernel for WINSHELD.EXE to register itself with the kernel.

- **USER.EXE**

  This module is based on the Windows 3.1 USER.EXE, with all the seamless support code in the WIN-OS/2 3.0 USER.EXE ported across to it. USER.EXE contains the user interface and windowing parts of WIN-OS/2 3.1.

- **GDI.EXE**

  This module is based on the Windows 3.1 GDI.EXE, and the palette management code from the previous WIN-OS/2 3.0 GDI.EXE has been ported across and improved. The printer spooler mapping that eliminates double spooling by Windows applications has also been ported across from the WIN-OS/2 3.0 GDI.EXE.

- **SYSTEM.INI and WIN.INI**

  These WIN-OS/2 3.1 initialization files are based on the Wilndows 3.1 files of the same name and have additional fields for WIN-OS/2 to specify parameters such as the full-screen and seamless display drivers, and the printer drivers.

- **WINSHELD.EXE**

  This WIN-OS/2 module has been enhanced with Windows 3.1 seamless and full screen support.

- **VWIN.SYS**

  This virtual device driver handles the synchronization of and communication between the WIN-OS/2 3.1 and OS/2 environments. Some of these tasks were formerly performed by the VDMSRVR.EXE process under OS/2 in WIN-OS/2 3.0 but this module has now been completely replaced.

Changes have also been made to some of the OS/2 modules as follows

- **PMVIOP.DLL**

  This DLL has been enhanced to support the new WIN-OS/2 3.1 settings, DOS_AUTOEXEC and WIN_RUNMODE.

- **SESMGR.DLL**

  Two new program types have been added to the Session Manager in order to support Windows 3.1 standard mode:

  - **PROG_31_STDSEAMLESSVDM** - Windows 3.1 program that will execute in its own windowed WIN-OS/2 session.
  - **PROG_31_STDSEAMLESSCOMMON** - Windows 3.1 program that will execute in a common windowed WIN-OS/2 session.

- **OS2KRNL**

  A new return value to DOSQAppType has been added to determine the Windows version (3.0 or 3.1) of the application. This value is used in the migration of Windows applications to the appropriate common VDM.

- **WPSH.DLL**

  The **Session** page of WIN-OS/2 objects, has been updated for WIN-OS/2 3.1 support. The Workplace Shell is able to detect if WIN-OS/2 3.1 is installed and display the updated session page.

7.4 WIN-OS/2 3.1 Enhanced Compatibility Mode

Windows 3.1 enhanced-mode applications which do not access VxDs (a specific type of Windows device drivers) can run under WIN-OS/2 3.1 in enhanced compatibility mode.

This includes applications such as Mathematica for Windows and Omnipage Professional.

This is not an implementation of Windows 3.1 386 enhanced mode, but a mode specific to WIN-OS/2 3.1 which enables major Windows enhanced-mode applications to run. OS/2 Version 2 users already enjoy many of the benefits of WIndows 3.1 enhanced mode, such as virtual memory, through the built-in functions of OS/2 Version 2.

7.4.1 Setting Up Applications for Enhanced Compatibility Mode

Enhanced mode Windows applications registered in the OS/2 Migration Database will be set up for WIN-OS/2 3.1 enhanced compatibility mode if the Migrate Applications Utility is used.

To verify this, or to manually set up an application to run in the WIN-OS/2 3.1 enhanced compatibility mode, follow these steps:

1. Display the pop-up menu of the application object by clicking the right mouse button on the application icon.
2. Display the **Open** options by clicking the left mouse button on the right pointing arrow within the **Open** box, as shown in Figure 53.

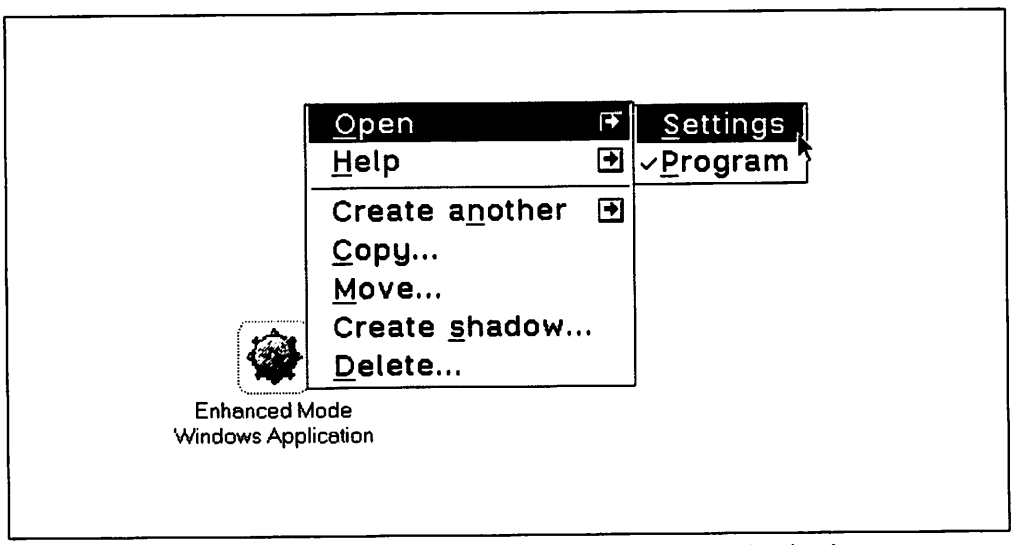


Figure 53. Enhanced Compatibility Mode: Accessing the Settings Notebook

3. Click the left mouse button on the **Settings** option and the **Settings** notebook for the application object will be displayed.
4. Click the left mouse button on the **Session** tab to jump to that page as shown in Figure 54 on page 82.

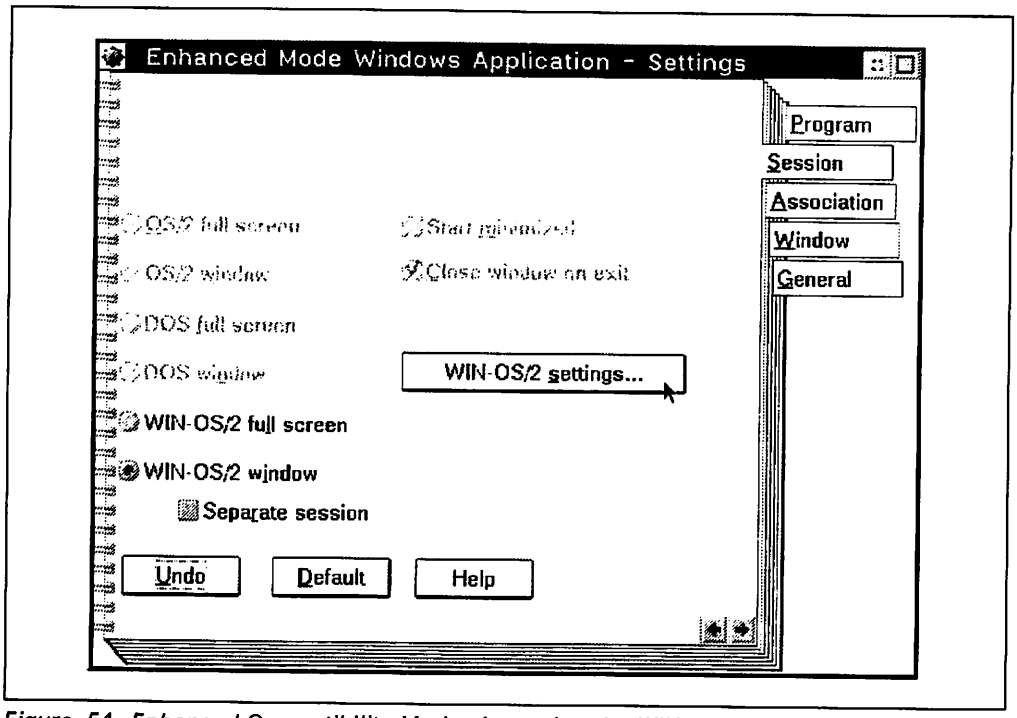


Figure 54. Enhanced Compatibility Mode: Accessing the WIN-OS/2 Settings

5. With either the radio button of the WIN-OS/2 full screen or the WIN-OS/2 window turned on, click the left mouse button on the **WIN-OS/2 settings** box and the **WIN-OS/2 Settings** panel will be displayed, as shown in Figure 55.

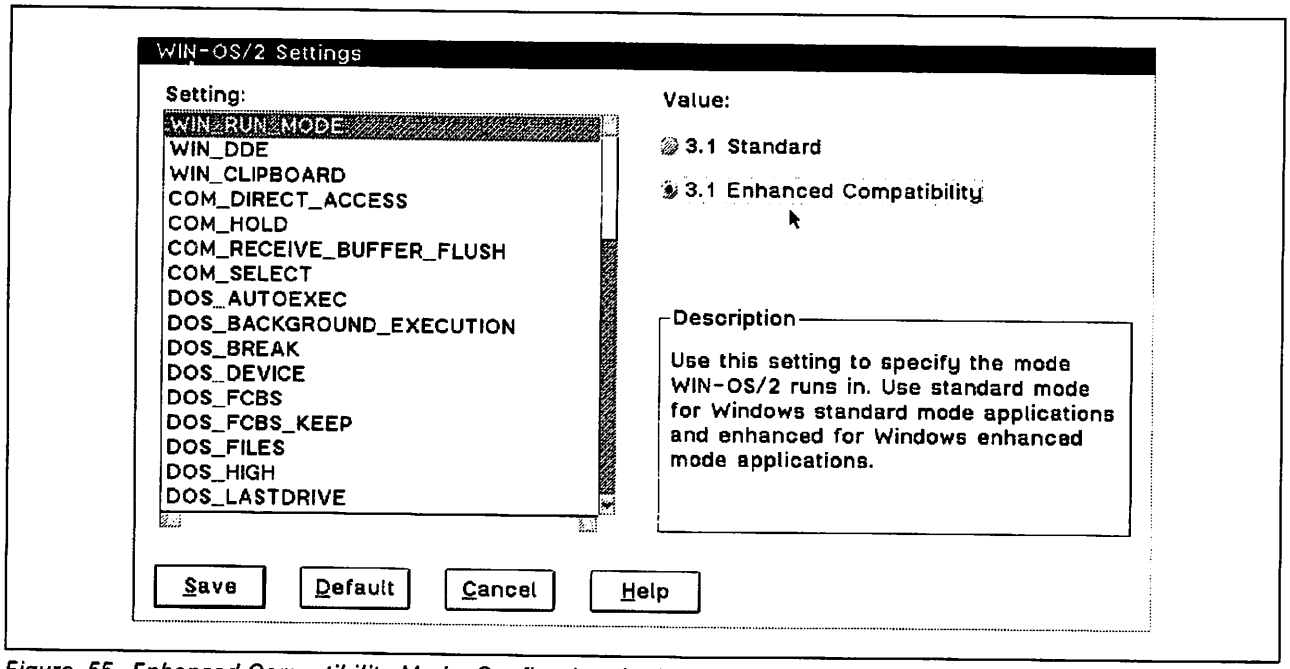


Figure 55. Enhanced Compatibility Mode: Configuring the Mode

6. With the **WIN_RUNMODE** list item highlighted, click the left mouse button on the **3.1 Enhanced Compatibility Mode** radio button to select the desired enhanced compatibility mode.

7. Click the left mouse button on the **Save** box to register the settings made and close the Settings notebook by double-clicking the left mouse button on the title bar icon.

7.4.2 Enhanced Compatibility Mode from the Command Line

To start an enhanced compatibility mode session from an OS/2 or DOS command line, type **WINOS2 /E** or **WINOS2 /3**.

Users may also start an application to run in the enhanced compatibility mode from the OS/2 or DOS command line by typing the application program name after the **WINOS2 /E** or **WINOS2 /3** command. These commands are not case-sensitive.

Example

```
[C:\] WINOS2 /E hello.exe

c:\> WINOS2 /E hello.exe
```

One way to check if the run mode is enhanced compatibility mode in a multiple applications VDM is to bring up the **Help** pop-up menu from the Program Manager and select **About....** This displays the **About Program Manager** box as shown in Figure 56, which will inform the user if the session is in standard mode or enhanced compatibility mode.

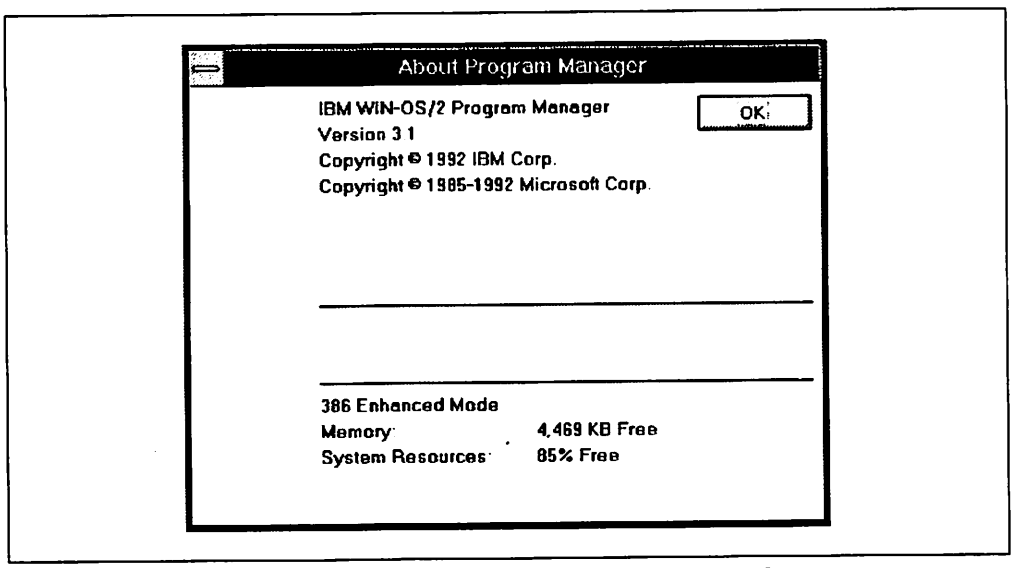


Figure 56. Enhanced Compatibility Mode: Program Manager About Box

Note

Although the WIN-OS/2 3.1 About box says 386 Enhanced Mode, what it really means is 386 Enhanced Compatibility Mode.

7.4.3 Implementation

WIN-OS/2 3.1 enhanced compatibility mode uses the same modules as WIN-OS/2 3.1 standard mode (unlike Windows 3.1, which uses a different kernels and other modules in order to provide virtual memory and other features already provided by OS/2 2.1).

To prevent any ill-behaved or defective Windows applications from disrupting the entire system, the WIN-OS/2 3.1 environment remains encapsulated. Like the previous release of OS/2 2.0, a special VDM is provided to emulate a DPMI server and the WIN-OS/2 3.1 kernel is loaded into the VDM to directly service the requests of Windows applications running in the VDM. To preserve the integrity of the system and avoid having duplicate virtual device drivers (one running on top of the other), the WIN-OS/2 3.1 enhanced compatibility mode does not use the Windows 3.1 386 enhanced mode virtual device drivers (VxDs).

Additional modifications for enhanced compatibility mode have been made to two components of WIN-OS/2, WINOS2.COM and OS2K386.EXE, and also to the Workplace Shell.

- **WINOS2.COM** and **OS2K386.EXE**

  The new WIN-OS/2 3.1 kernel, OS2K386.EXE, interacts with WINOS2.COM and the enhanced mode Windows application to reply to the Windows application that it is running in enhanced mode. This approach may not work for all Windows enhanced mode applications, but enables Mathematica and some other enhanced mode Windows applications to run.

- **WPSH.DLL**

  As described in 7.3, "WIN-OS/2 3.1 Standard Mode" on page 77, PMVIOP.DLL has been modified to provide two radio buttons for selection of either 3.1 standard mode or 3.1 enhanced compatibility mode in the WIN_RUNMODE option.

  Three new program types have also been added to the session manager, SESMGR.DLL, to support WIN-OS/2 3.1 enhanced compatibility mode:

  - **PROG_31_ENHSEAMLESSVDM** - Windows 3.1 program that will execute in enhanced compatibility mode in its own windowed WIN-OS/2 3.1 session.
  - **PROG_31_ENHSEAMLESSCOMMON** - Windows 3.1 program that will execute in enhanced compatibility mode in a common windowed WIN-OS/2 3.1 session.
  - **PROG_31_ENH** - Windows 3.1 program that will execute in enhanced compatibility mode in a full screen WIN-OS/2 3.1 session.

7.5 Improved OLE Support

Object Linking and Embedding (OLE) enables compound documents to be created, modified and printed. Compound documents are documents created using more than one application - for example a letter with embedded graphics.

OLE defines two approaches to compound documents:

Embedding - the compound document contain sections "embedded" in the main document, which have been created by another application. For example, a letter created by a word-processing application could

contain a diagram created by a drawing application. There is only one document, but by double-clicking on the embedded section the second application is invoked to process the section.

Linking - the compound document consists of one document with links to part or all of other documents which have been created by another application. The linked documents can be processed separately, or as part of the compound document. Changes to the linked documents are reflected in the compound document when it is processed

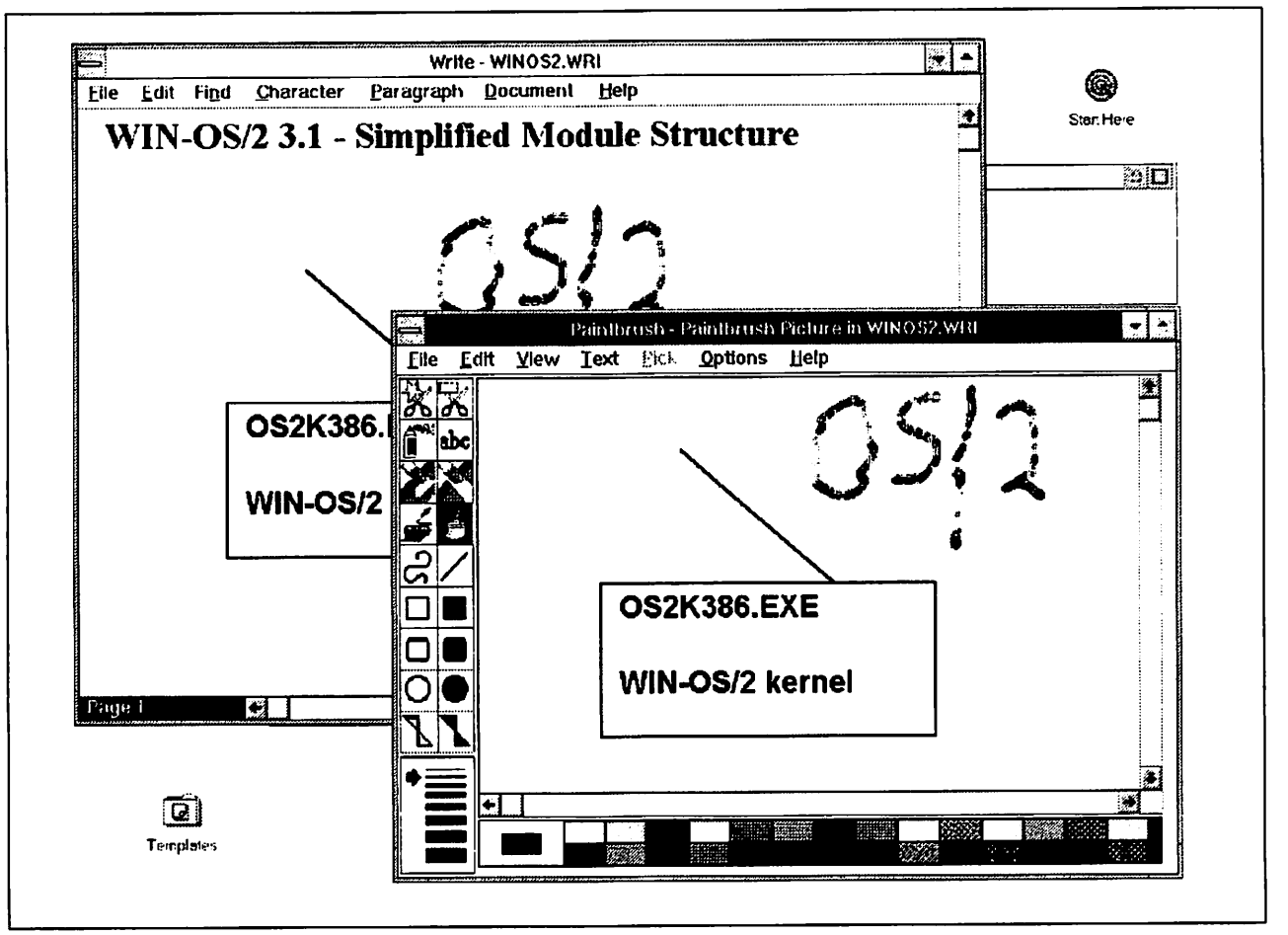


Figure 57. Windows Applications Using OLE under WIN-OS/2 3.1

OLE implementation needs both system support, and also application support (from at least two applications, for obvious reasons).

In OS/2 2.1, OLE system support is supported in WIN-OS/2 3.1 within the same Windows session - either a MAVDM or a common seamless session. Application support is provided in a number of common Windows applications, including the Windows applets Write and Paint Brush which are provided as part of WIN-OS/2 3.1.

7.6 Multimedia Support for Audio

Windows 3.1 included audio multimedia support as part of the base product. This is also supported in WIN-OS/2 3.1 in OS/2 2.1.

As with MMPM/2, to use these multimedia features, special hardware is required - for example, a CD-ROM drive, or an audio card.

Two Windows multimedia applets are included with WIN-OS/2 3.1:

Media Player provides an interface for controlling various multimedia hardware, in conjunction with the appropriate device driver. For example, the Media Player can be used to play a Beatles CD through headphones attached to the phono socket.

Sound Recorder works in conjunction with an audio card to enable digital sound recording and playback - for example for voice messages.

7.7 ATM 2.5 and TrueType Font Support

Windows 3.1 included TrueType font support. TrueType is an advanced font technology - similar in concept to ATM - which enables scalable fonts to be accurately displayed and printed.

WIN-OS/2 3.1 includes both TrueType and ATM font technologies. TrueType is installed by default. However, since ATM is also implemented in OS/2 2.1 PM, we suggest that ATM is used in preference to TrueType since it simplifies moving documents and clipboard data between WIN-OS/2 and Presentation Manager.

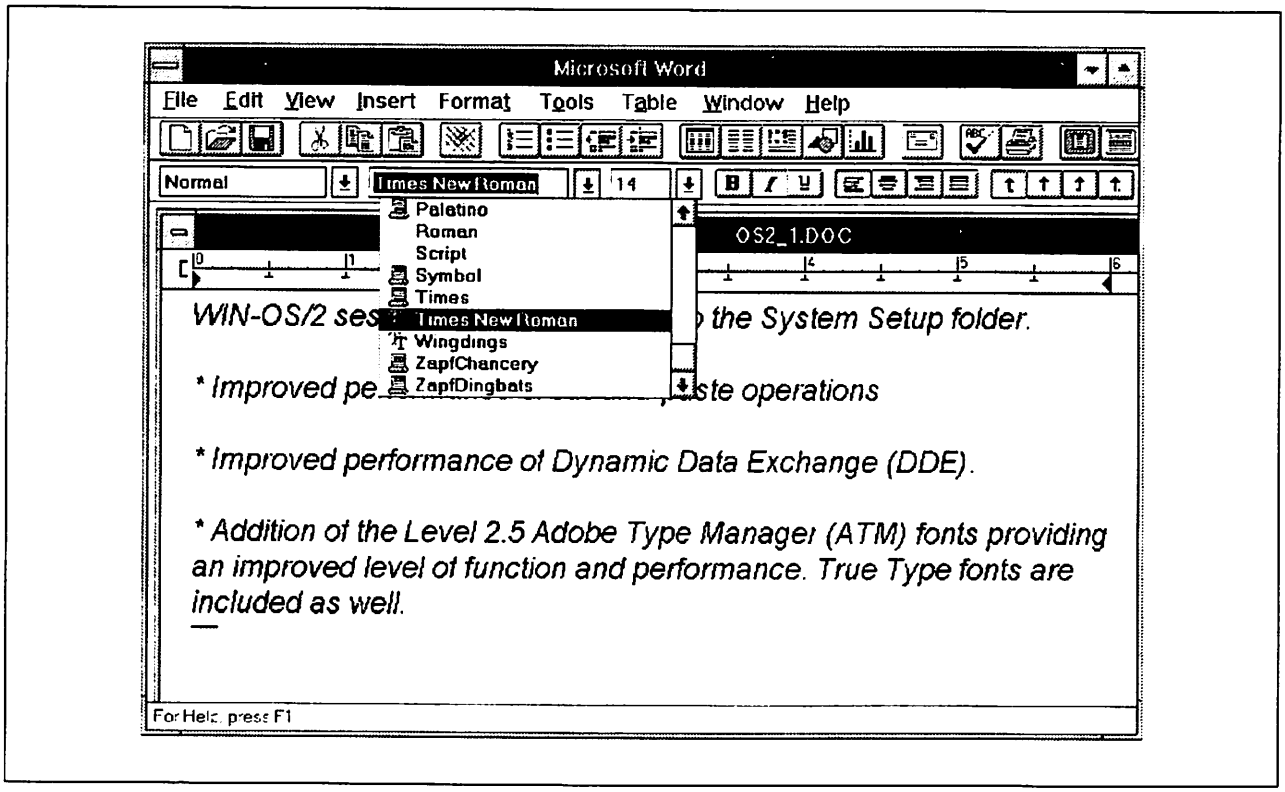


Figure 58. Selecting a TrueType Font

In order to maintain compatibility for Windows 3.1 applications, TrueType font support and the base TrueType fonts are included in WIN-OS/2 3.1.

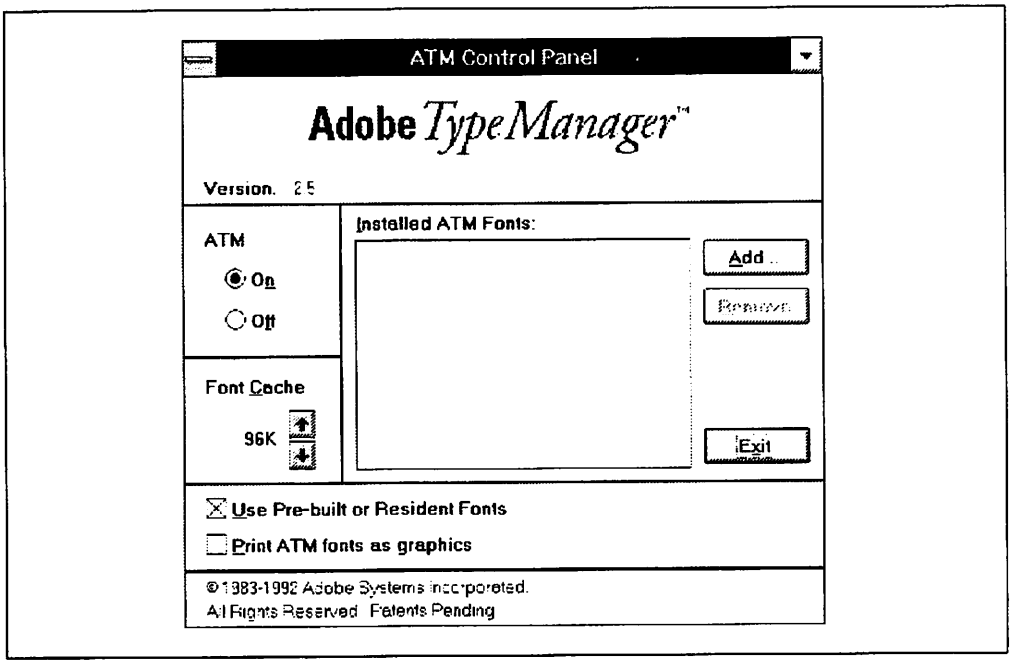


Figure 59. WIN-OS/2 3.1 ATM Control Panel

Note

The WIN-OS/2 3.1 ATM fonts are not installed by default, but can be installed from the printer driver diskettes using the ATM Control Panel from within WIN-OS/2 3.1.

7.8 Starting DOS and OS/2 Applications from WIN-OS/2 3.1

Some Windows applications rely on being able to call DOS utility programs for basic and utility functions.

WIN-OS/2 3.0 did not include the ability to start non-Windows applications from a VDM running WIN-OS/2.

With the new WIN-OS/2 3.1 support in OS/2 2.1, users are now able to start DOS and OS/2 applications from a VDM running a WIN-OS/2 3.1 full screen or seamless session. There may be some exceptions in terms of which DOS utilities can be launched.

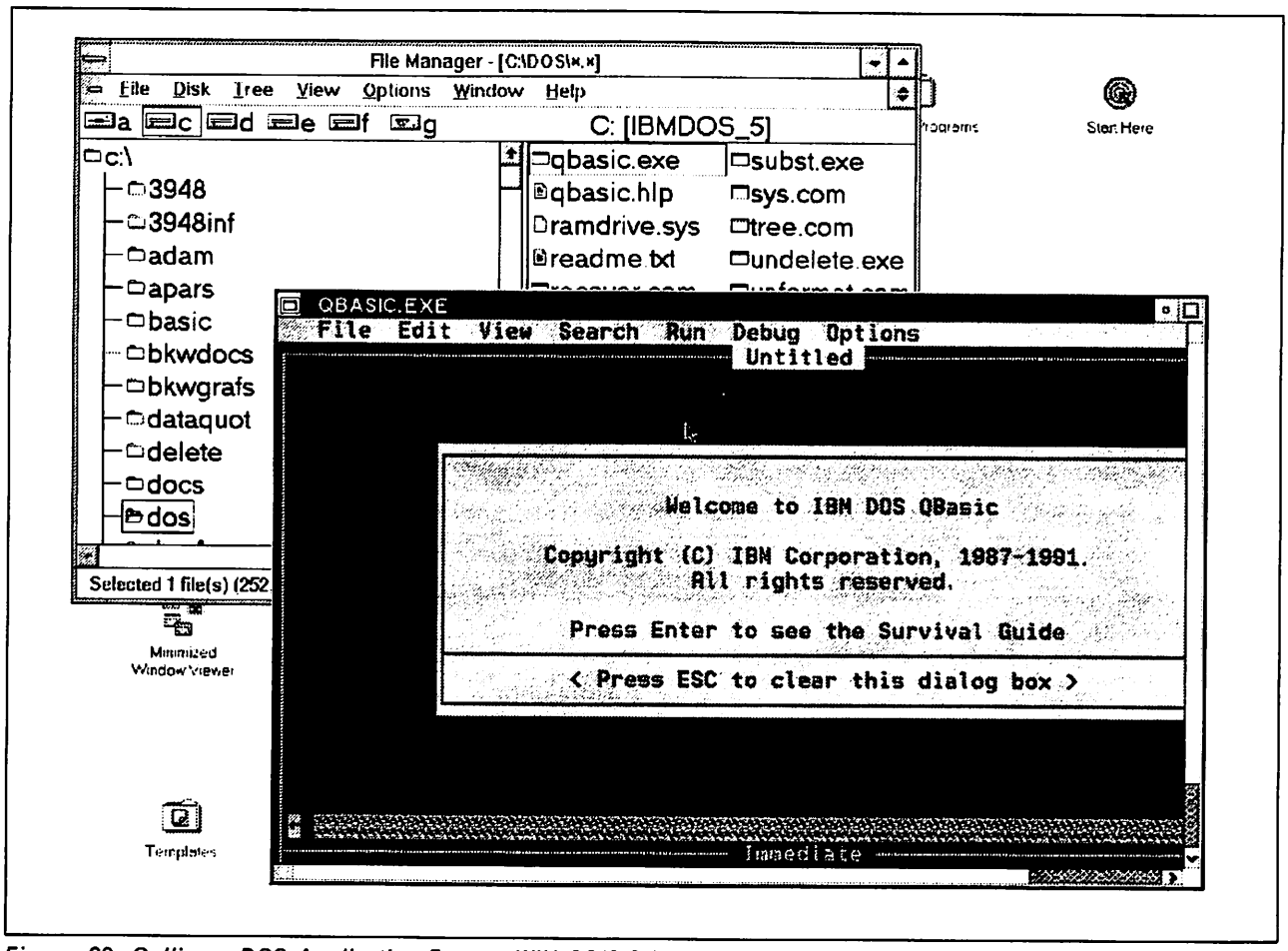


Figure 60. Calling a DOS Application From a WIN-OS/2 3.1 Application

Modifications have been made to the WIN-OS/2 kernel (OS2K386.EXE), DOS EMulation (DEM), Multiple Virtual DOS Machines (MVDM) and Session Manager (SESMGR.DLL) in order to achieve this kind of application launching.

- **OS2K386.EXE**

  This new WIN-OS/2 kernel now has the ability to detect non-Windows applications and call DOS EMulation (DEM).

- **DEM**

  A new API, DosStartSession, has been included. There is an existing INT 21h in DEM which provides services to some OS/2 APIs. DosStartSession has been added to this list, in order to provide an interface between DEM and SESMGR.DLL.

- **MVDM**

  The MVDM code servicing the INT 21h request has been enhanced.

- **SESMGR.DLL**

  A new thread has been used to interface with DEM.

7.9 WIN-OS/2 3.1 Display Drivers

In OS/2 2.0, seamless WIN-OS/2 support was only available for VGA. Full screen Windows support was provided for CGA, EGA, VGA, XGA and 8514 adapters. Support was also provided to allow DOS full screen and WIN-OS/2 full screen sessions to use the display system in SVGA mode, while Presentation Manager used the display as a VGA device.

With OS/2 2.1, it is now possible to run Windows applications seamlessly on XGA, XGA-2, SVGA, 8514/A, and VGA displays.

In addition, the previous full screen Windows support continues to be provided for CGA and EGA displays.

Two WIN-OS/2 3.1 display drivers are provided for each video configuration, one for full-screen WIN-OS/2, and one for seamless WIN-OS/2.

The full-screen WIN-OS/2 3.1 display driver is similar to a Windows 3.1 display driver, but includes some modifications such as allowing the WIN-OS/2 session to run in the background but without writing to the screen.

The seamless WIN-OS/2 3.1 display drivers also needs to handle running in a window - hence the display area is variable, and updates need to be coordinated with Presentation Manager via the WINSHELD module.

For a more detailed discussion of the SVGA and XGA display drivers and of the installation steps involved, please refer to 4.9, "Video Support" on page 53 and 3.3, "Display Driver Install Program" on page 30. The *IBM OS/2 Device Driver Development Kit* also contains detailed descriptions of the display driver architecture.

7.10 WIN-OS/2 3.1 Printer Drivers

WIN-OS/2 3.1 improves the support for Windows printer drivers. Some new printer drivers for Windows 3.1 have been modified in order to work under WIN-OS/2 3.1 As a result, a number of additional Windows printer drivers have been packaged with OS/2 2.1. For a more detailed description of the Windows printer drivers and the installation steps involved, please refer to Chapter 10, "Print Subsystem Enhancements" on page 129.

7.11 DDE and Clipboard Enhancements

The Clipboard is a temporary storage area for user-initiated data transfers between applications. Dynamic Data Exchange (DDE) is a protocol and a set of functions that enable applications to exchange data through program-to-program communication. These facilities have become increasingly important since the release of OS/2 2.0, both in mixed environments of Windows and OS/2 PM applications, as well as between Windows applications and between OS/2 PM applications.

A new Workplace Shell object for the global WIN-OS/2 Settings has been added to the system. It is represented by a new icon, WIN-OS/2 Setup, in the System Setup folder which is kept in the OS/2 System folder on the desktop. A separate object has been created, rather than using the System object, since these

settings are specific to the WIN-OS/2 environment, and also to avoid crowding too many settings into the System object.

The Clipboard Viewers provided in both the WIN-OS/2 and PM are still available for users to view Clipboard data, but it is no longer necessary to run the Clipboard Viewers in order to exchange data between PM and WIN-OS/2 applications.

7.11.1 Setting Clipboard and DDE to Public or Private

New user interfaces have been designed to enable users to specify either public mode or private mode for the WIN-OS/2 3.1 Clipboard and DDE. By default, both the Clipboard and DDE, for both OS/2 and WIN-OS/2, are public and users can exchange data between programs running in DOS, OS/2 and WIN-OS/2 sessions. If the WIN-OS/2 Clipboard or DDE is set to private, data exchange is only allowed among programs in the same WIN-OS/2 session.

Configuring the Clipboard and DDE as public or private is independent of whether sessions are common or separate.

Users can select the WIN-OS/2 operating mode of Clipboard and DDE for all WIN-OS/2 sessions through a new global WIN-OS/2 Setup object in the System Setup folder of the Workplace Shell, or for a particular WIN-OS/2 separate session using the new settings in the WIN-OS/2 settings notebook page.

To make the Clipboard and DDE public or private for all WIN-OS/2 sessions, follow these steps:

1. On the OS/2 PM desktop, double click the left mouse button on the **OS/2 System** icon and a System Setup folder icon appears.
2. Double click the left mouse button on the **System Setup** folder icon and a new **WIN-OS/2 Setup** icon appears as shown in Figure 61.

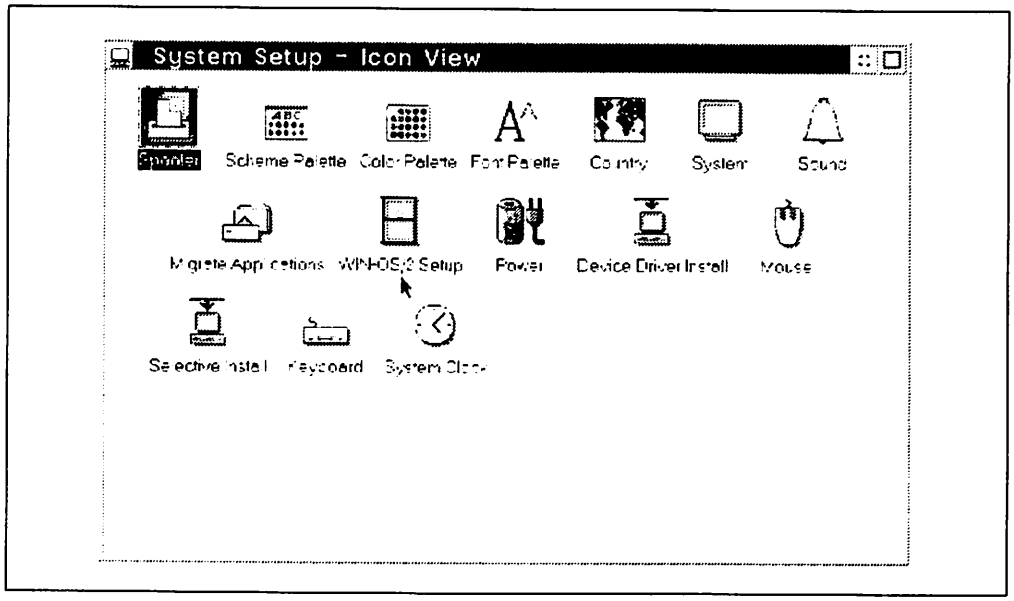


Figure 61. WIN-OS/2 3.1 Setup Object

3. Double click the left mouse button on this **WIN-OS/2 Setup** icon to run the program, and the **Settings** notebook for WIN-OS/2 Setup is displayed as shown in Figure 62 on page 91.

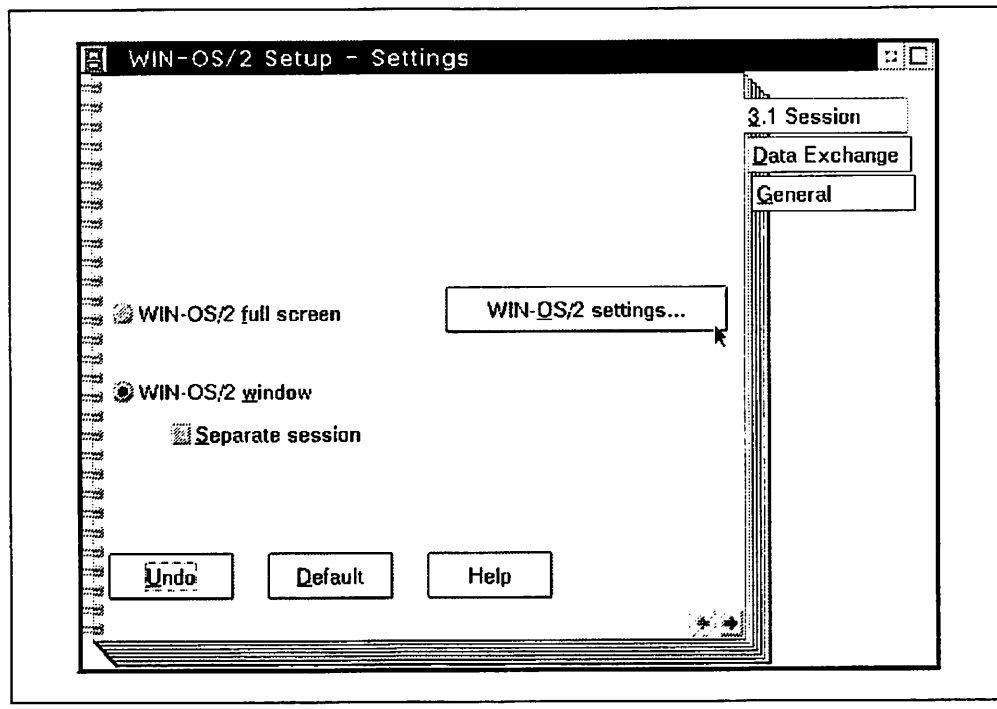


Figure 62. WIN-OS/2 3.1 Setup - 3.1 Session Definition

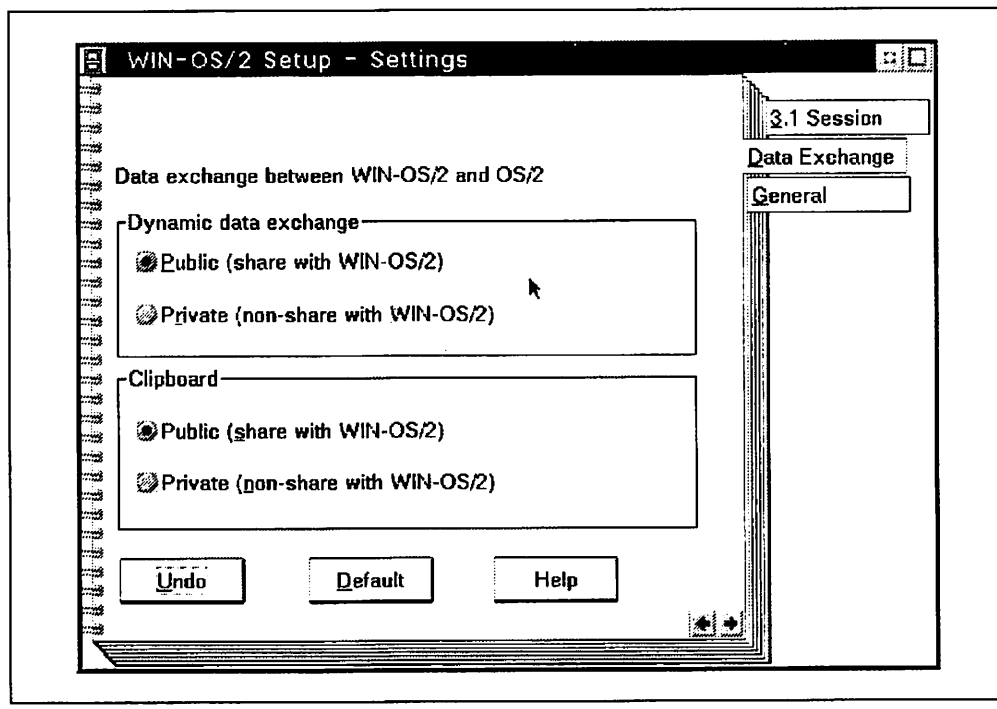


Figure 63. WIN-OS/2 3.1 Setup - Data Exchange

4. The user can now set the data exchange mode (**Public** or **Private**) between PM and WIN-OS/2 applications by clicking on the respective radio buttons.

Note

Any selection made on this page will affect all subsequent OS/2 and WIN-OS/2 3.1 sessions.

5. Finally, close the WIN-OS/2 Setup **Settings** notebook by double clicking the left mouse button on the title bar icon.

To make the WIN-OS/2 Clipboard and DDE public or private for a particular WIN-OS/2 separate session, follow these steps:

1. Open the WIN-OS/2 Program folder which contains the desired object by double clicking the left mouse button on the folder icon.
2. Display the pop-up menu of the program object by clicking the right mouse button on the program icon.
3. Display the **Open** options by clicking the left mouse button on the right pointing arrow within the **Open** box.
4. Click the left mouse button on the **Settings** option, and a **Settings** notebook for the program object will be displayed.
5. Click the left mouse button on the **Session** tab to jump to that page as shown in Figure 64.

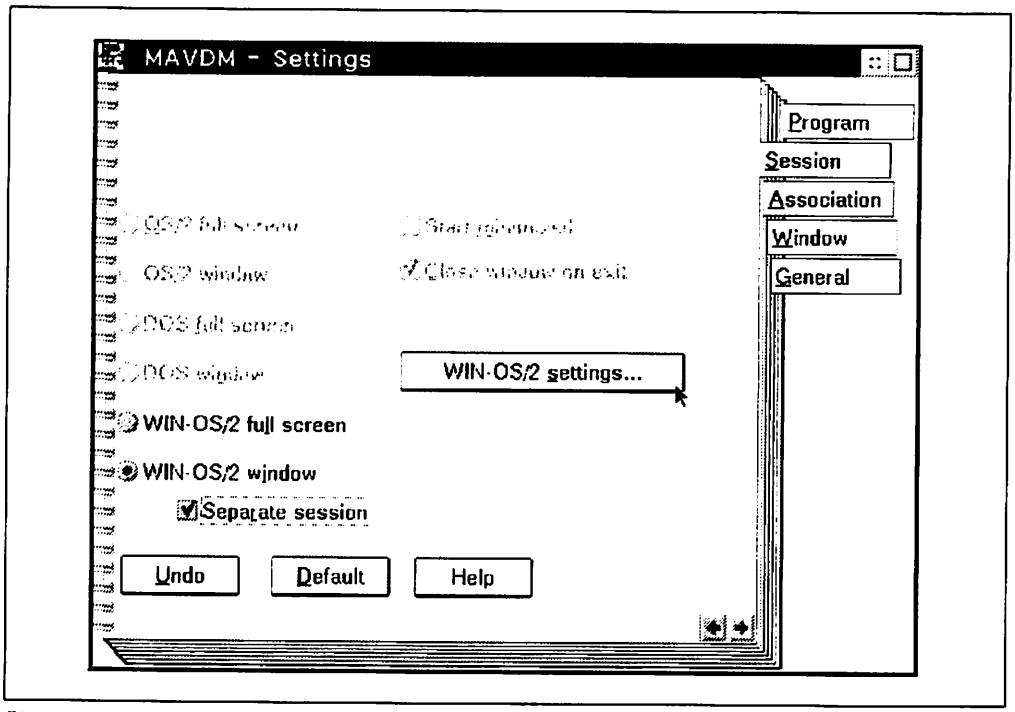


Figure 64. MAVDM Settings - Session Page

6. Click the left mouse button on the **WIN-OS/2 settings** box to open up the **WIN-OS/2 Settings** panel.
7. Click the left mouse button on the **WIN_DDE** or **WIN_CLIPBOARD** item to change the operating mode of the desired facility and the selected setting will be highlighted. The respective value and a brief description of the setting will be displayed as shown in Figure 65 on page 93 and Figure 66 on page 93.

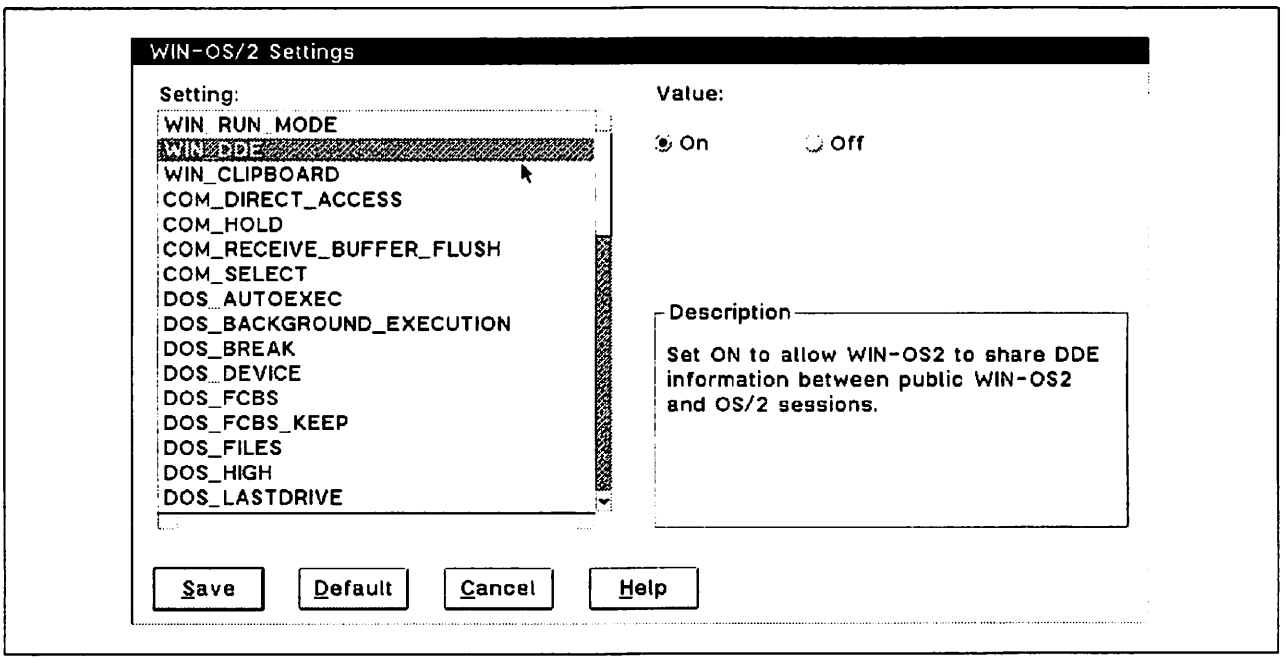


Figure 65. WIN-OS/2 3.1 Settings: WIN_DDE

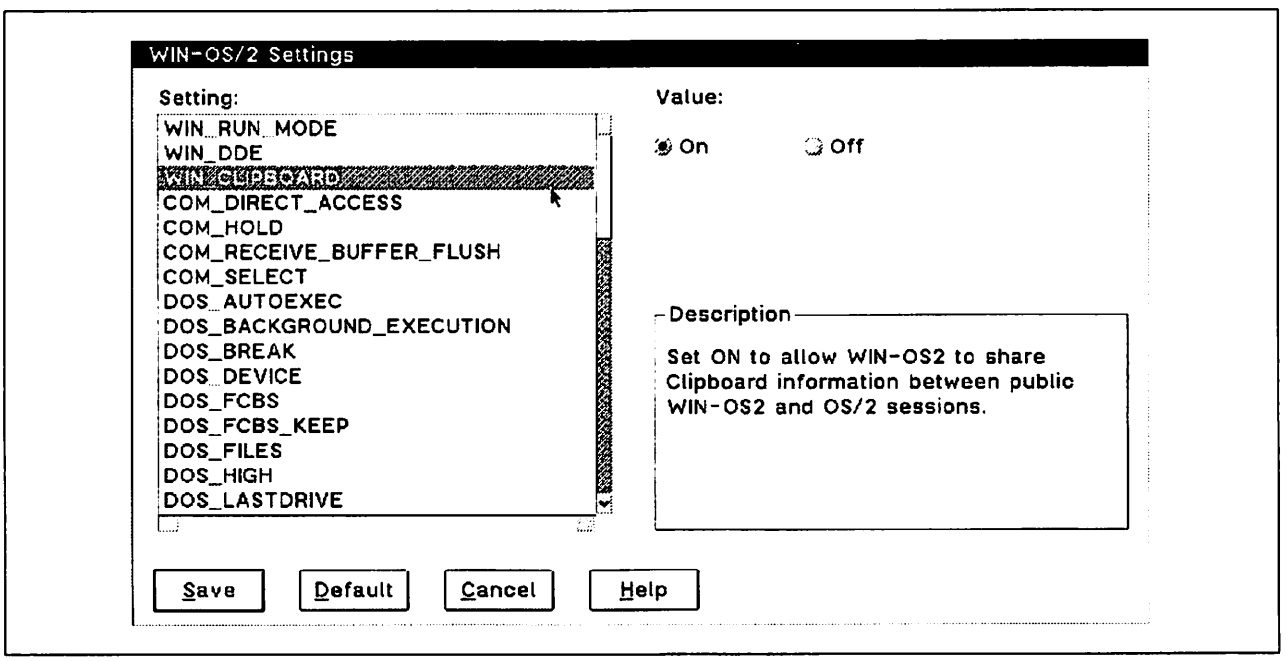


Figure 66. WIN-OS/2 Settings: WIN_CLIPBOARD

8. Click the left mouse button on the radio button of the desired setting value and click the left mouse button on the **Save** box to register the value chosen.
9. Close the program object **Settings** notebook by double-clicking the left mouse button on the title bar icon.
10. The program may now be restarted by double clicking the left mouse button on the object icon.

7.11.2 Implementation

The Clipboard and DDE implementations in WIN-OS/2 3.1 have been redesigned in order to run faster and to use less system resources. These improvements have been achieved by changing the communications technique between PM and VDMs from named pipes to a Virtual Device Driver (VDD) interface. Named pipes allow related or unrelated processes to communicate with each other. As the usage of Clipboard and DDE increases, the communications traffic becomes substantially higher. This high volume of traffic can affect the performance on systems which might already utilize named pipes heavily for other purposes or when the content of the clipboard data is very large (such as bitmaps).

The PM Clipboard and DDE programs have also been modified to be threads under PMVIOP.DLL rather than separate executable processes (as in WIN-OS/2 3.0).

The Clipboard and DDE programs for WIN-OS/2 have been modified to be window procedures under the WIN-OS/2 Shield (WINSHELD.EXE) rather than separate executable programs. This consolidation means that the icons for these programs have vanished from the desktop.

When there is a change in mode between public and private, PMVIOP.DLL is notified of the changes through a new API, DDEClipNotify(). When there is a change in the PM settings, PMVIOP.DLL will start or stop the Clipboard or DDE threads as necessary. These threads perform the required operations to update the operating mode using VWIN.SYS for communications between processes. If the WIN-OS/2 setting is changed, PMVIOP.EXE will send a message through VWIN.SYS to notify WINSHELD.EXE of the change. WINSHELD.EXE then starts or stops the appropriate window procedures for Clipboard or DDE.

The following modules have been modified to implement the VDD interface for communications used in the Clipboard and DDE, in order to provide data exchange between applications.

- **VWIN.SYS**

  Changes have been made to this VDD to perform the functions previously provided by VDMSRVR.EXE in WIN-OS/2 3.0. The virtual device driver VWIN.SYS accepts registration from the Clipboard and DDE programs in both the WIN-OS/2 and PM environments. VWIN was already used as the communications mechanism for seamless windows support.

- **PMVIOP.DLL**

  Changes have been made to this DLL to spawn threads for Clipboard and DDE. These threads are very similar to the previous Clipboard and DDE programs for PM, except that the user interface has been removed. These threads use the new VWIN.SYS VDD communications interface rather than the named pipe interface.

- **WINSHELD.EXE**

  Changes have been made to WINSHELD.EXE to create window procedures for Clipboard and DDE. These window procedures are very similar to the previous Clipboard and DDE programs for WIN-OS/2, except that the user interface has been removed for Clipboard and DDE. As with PMVIOP.DLL, these use the new VWIN.SYS VDD communications interface rather than the named pipe interface. This eliminates the dependency on VDMSRVR.EXE which is removed from the system.

- **CLIPBRD.EXE (renamed from CLIPWOS2.EXE)**

  The WIN-OS/2 Clipboard has been modified to remove the code that interfaces with VDMSRVR.EXE (this is now performed by WINSHELD.EXE). This program is now only a Clipboard Viewer and it has been renamed to CLIPBRD.EXE. It is still kept in the same directory (\OS2\MDOS\WINOS2).

- **CLIPOS2.EXE**

  The OS/2 Version 2 Clipboard has also been modified to remove the code that interfaces with VDMSRVR.EXE (this is now performed by PMVIOP.DLL). This program is now only a Clipboard Viewer and it remains in the same directory (\OS2\APPS).

VDMSRVR.EXE (the previous global Clipboard and DDE server), DDEAGENT.EXE and PMDDE.EXE modules have been removed from the system as their functions are now performed by other modules.

The data flow among these components is illustrated by the following figure:

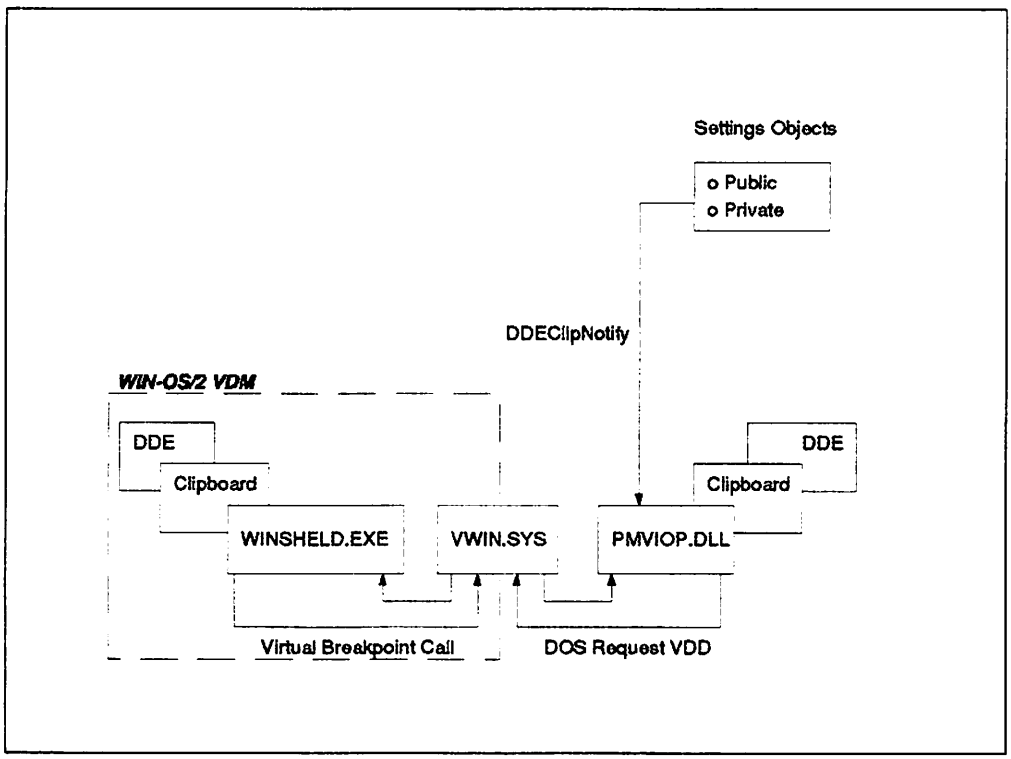


Figure 67. Clipboard and DDE Data Flow between WIN-OS/2 3.1 and OS/2 PM

7.12 Windows 3.1 Utilities and Accessories

A number of benefits are provided automatically through the inclusion of Windows 3.1 code into WIN-OS/2 3.1.

Many of the Windows 3.1 accessories are now included with WIN-OS/2 3.1, thus enabling users to be more productive with the WIN-OS/2 environment right away. Users can also enjoy the additional benefits from the improved OLE support and multimedia support for audio.

Windows 3.0 versions of Control Panel, Print Manager, Clipboard Viewer, ATM Control Panel, and the Clock were also included with WIN-OS/2 3.0. All the other Windows applets are shipped for the first time with WIN-OS/2 3.1.

The following Windows 3.1 utilities are now included with the WIN-OS/2 3.1 support in OS/2 2.1:

- File Manager
- Control Panel
- Print Manager
- Clipboard Viewer
- Win-OS/2 Setup
- ATM** Control Panel
- ATM ReadMe *(this is a text file on a Note Pad that contains some useful information on Adobe** Type Manager under WIN-OS/2 3.1)*

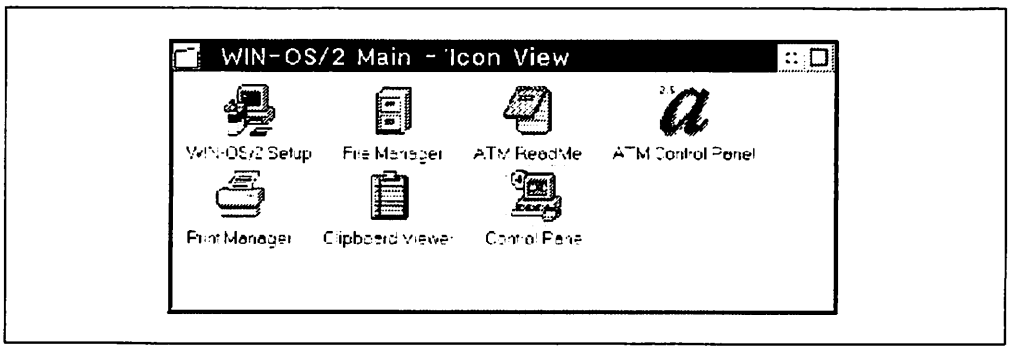


Figure 68. WIN-OS/2 3.1 - Utility Applications

The following Windows 3.1 applets are now included with WIN-OS/2 3.1 in OS/2 2.1:

- Write
- Paint Brush
- Calculator
- Clock
- Sound Recorder *(use the drivers option in the Control Panel to install sound drivers)*
- Cardfile
- Calendar
- Object Packager
- Character Map
- Notepad
- Media Player *(use the drivers option in the Control Panel to install MCI device drivers)*

A new Startup Group has been added for users to put in applications that they want to start automatically under WIN-OS/2 3.1.

The Windows 3.1 games and bitmaps, as well as the Windows Recorder and Terminal accessories, have not been included with WIN-OS/2 3.1.

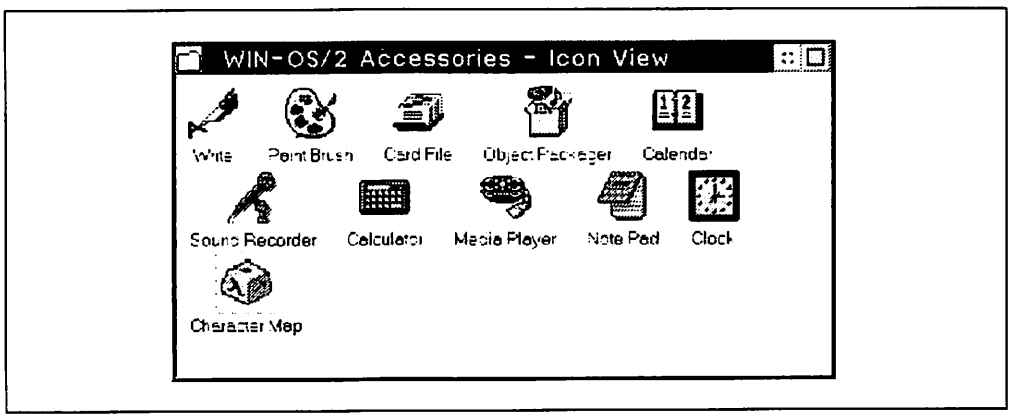


Figure 69. WIN-OS/2 3.1 - Accessories

7.13 Application Considerations

We recommend the use of the OS/2 2.1 Migrate Applications utility once the Windows application has been installed. This sets up the WIN-OS/2 settings correctly for the application, as well as creating an icon in the Windows Application folder.

7.13.1 Different Ways of Running WIN-OS/2 3.1 Applications

There are many different ways a WIN-OS/2 application can run, and in order to clarify this it is helpful to look at these ways from two different perspectives. One is from the "Session" point of view, and the other is from the "Settings" point of view.

7.13.1.1 WIN-OS/2 Sessions

Figure 70 shows a simplified view of the various WIN-OS/2 sessions.

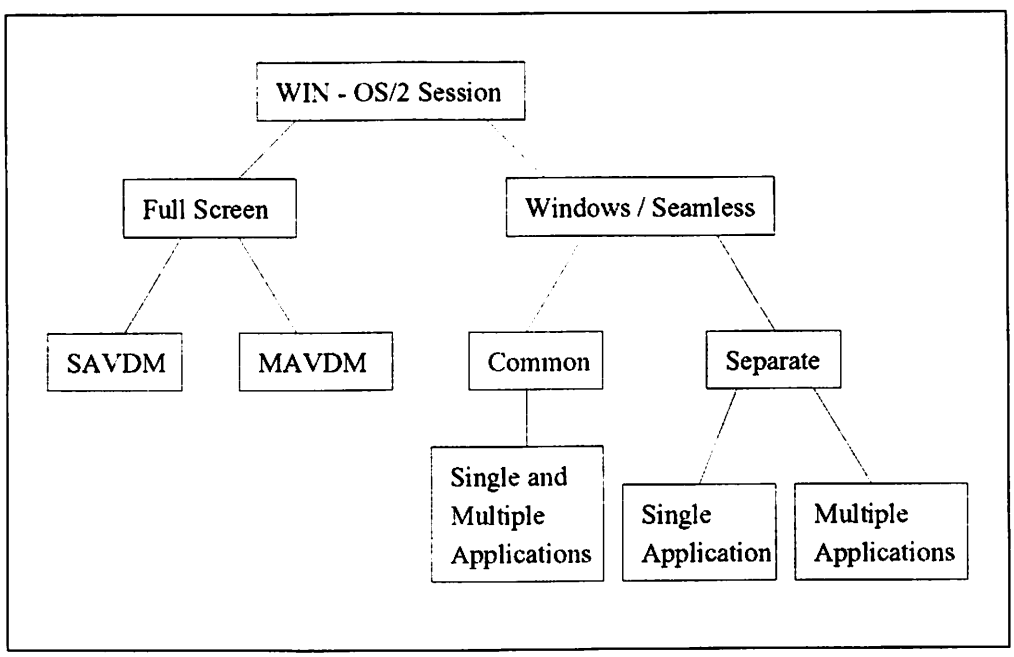


Figure 70. Different WIN-OS/2 Program Object Settings

A WIN-OS/2 session can be either full screen or seamless (windowed).

For every full screen WIN-OS/2 session, one Virtual DOS Machine (VDM) is started. If it runs just one application, it is called a Single Application VDM (SAVDM). If it runs more than one application (and it achieves this by using the Windows program manager PROGMAN.EXE), it is called a Multiple Application VDM (MAVDM). In both cases, there is only one WIN-OS/2 session running, in a single VDM.

A seamless WIN-OS/2 session can be either a separate seamless VDM or a common seamless VDM. Like the full screen WIN-OS/2 session, a separate seamless VDM can run either a single application, or multiple applications (using PROGMAN.EXE). There is only one WIN-OS/2 session, running in a single VDM. But unlike the full screen WIN-OS/2 session, a separate seamless VDM runs its applications seamlessly on the workplace shell desktop.

In a common seamless VDM, there can be one or more WIN-OS/2 sessions running seamlessly in a single VDM. Each WIN-OS/2 session can run one or more applications. There is only one common seamless VDM in the entire system and all seamless WIN-OS/2 sessions that are not defined as separate will run in this common seamless VDM.

Note

The first seamless WIN-OS/2 session launched that is not separate will set up the common seamless VDM with its own environment settings, and all subsequent common seamless VDM sessions will share this environment. Please take this into consideration when deciding which common seamless VDM WIN-OS/2 session to start first and ensure that the environment that this session sets up is appropriate for the entire VDM.

See 13.5.4.2, "WIN-OS/2 3.1 Memory Management Considerations" on page 203 for a detailed discussion of the implications of this for WIN-OS/2 memory usage.

7.13.1.2 WIN-OS/2 Program Object Settings

Windows applications should normally be installed on the Workplace Shell by using the Migrate Applications utility, since this will tailor the WIN-OS/2 settings from information in the migration database.

However, it is also possible to define a new WIN-OS/2 application on the Workplace Shell by creating a new program object from the Templates folder. Users may also make a copy from any existing program object, but remember to check through the settings inherited and change them appropriately. Figure 71 on page 99 shows a structured summary of WIN-OS/2 program object settings.

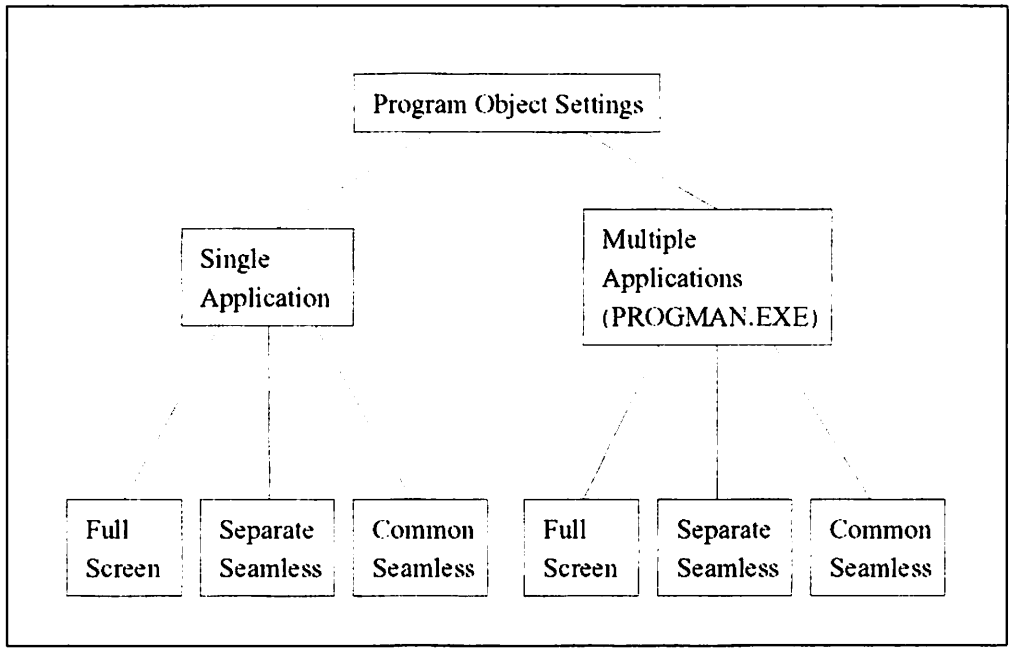


Figure 71. Different WIN-OS/2 Program Object Settings

Follow these steps to bring up the program object settings notebook:

1. Display the pop-up menu of the program object by clicking the right mouse button on the program icon.
2. Display the **Open** options by clicking the left mouse button on the right pointing arrow within the **Open** box.
3. Click the left mouse button on the **Settings** option and a **Settings Notebook** for the program object will be displayed as shown in Figure 72.

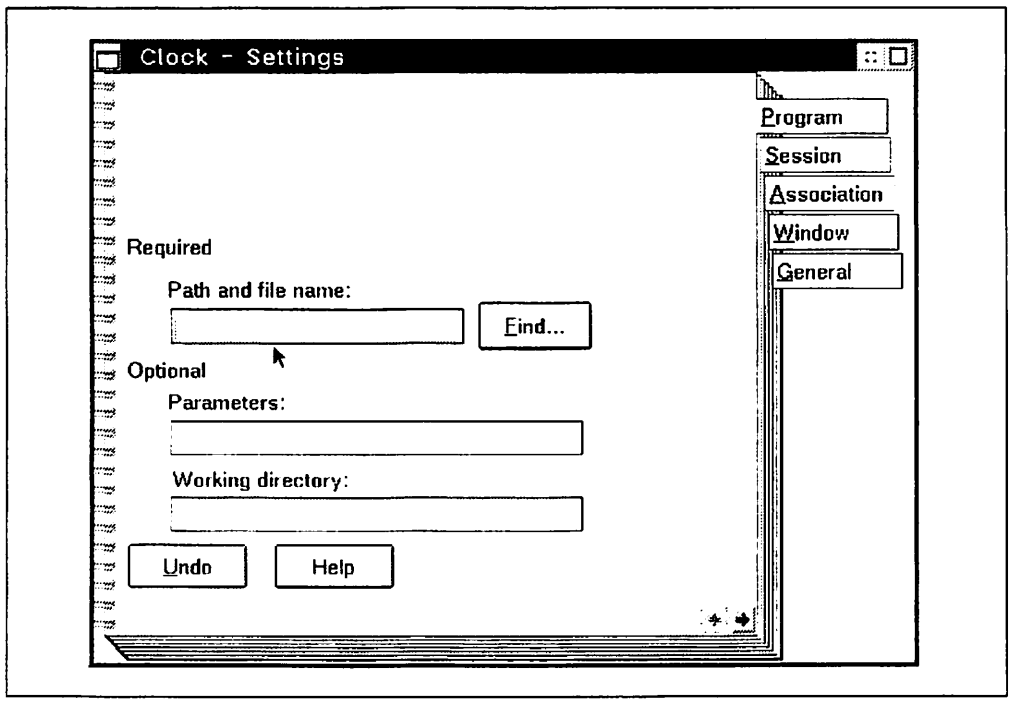


Figure 72. WIN-OS/2 3.1 - Program Object Settings Notebook

There are two scenarios for the program object definition, one for the single application environment and the other for the multiple applications environment.

For a single application definition, type in the name of the application program in the **Path and file name** field as shown in Figure 73.

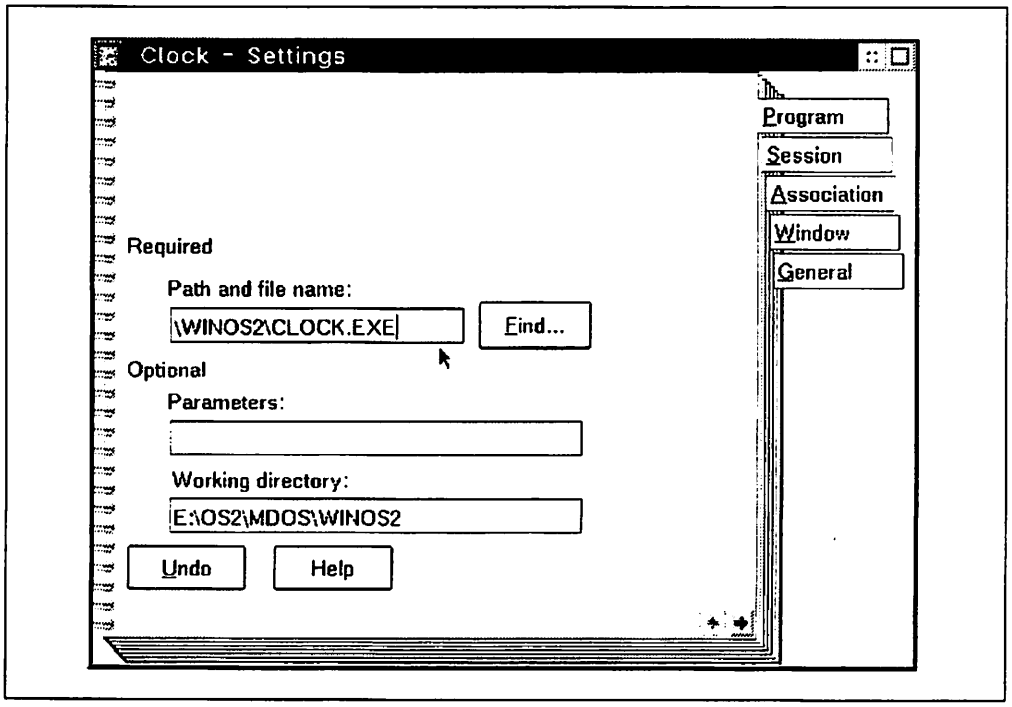


Figure 73. WIN-OS/2 3.1 - Defining a Single Application Program Object

For a multiple applications definition, type **PROGMAN.EXE** in the **Path and file name** field, and the names of the application programs (separated by commas) in the **Parameter** field. An example of defining three applications is shown in Figure 74 on page 101.

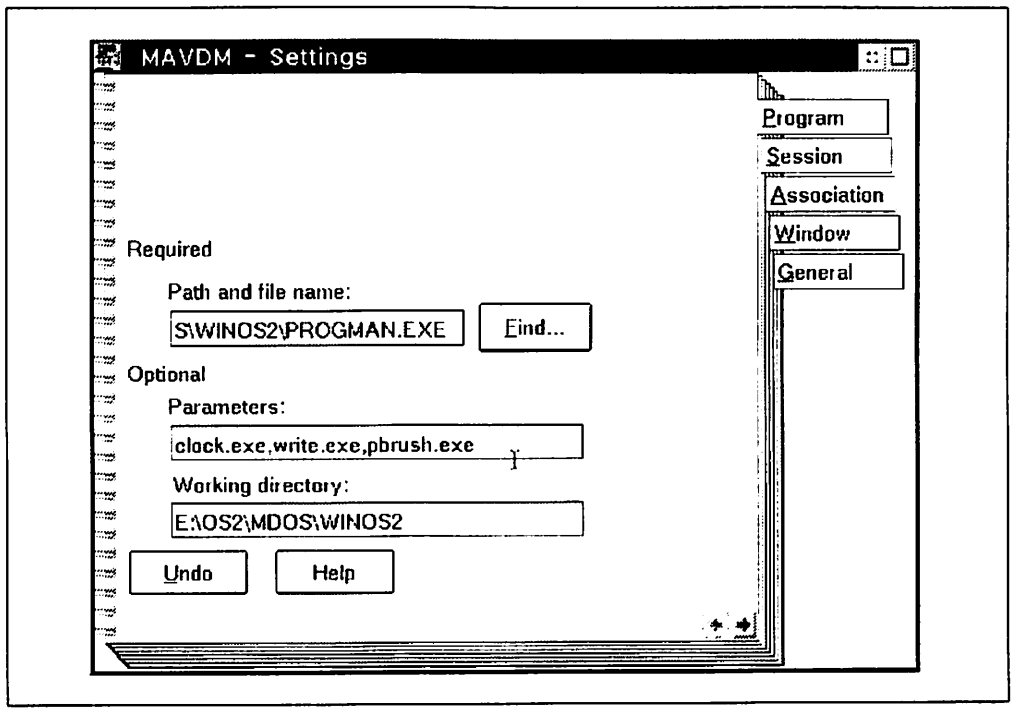


Figure 74. WIN-OS/2 3.1 - Defining a Multiple Applications Program Object

Once the Program definition has been done, go to the **Session** definition by clicking the left mouse button on the **Session** tab. For both the single application and multiple applications environments, three different types of WIN-OS/2 sessions can be set.

The first type is the WIN-OS/2 full screen session. This is selected by clicking the left mouse button on the radio button next to the **WIN-OS/2 full screen** option as shown in Figure 75.

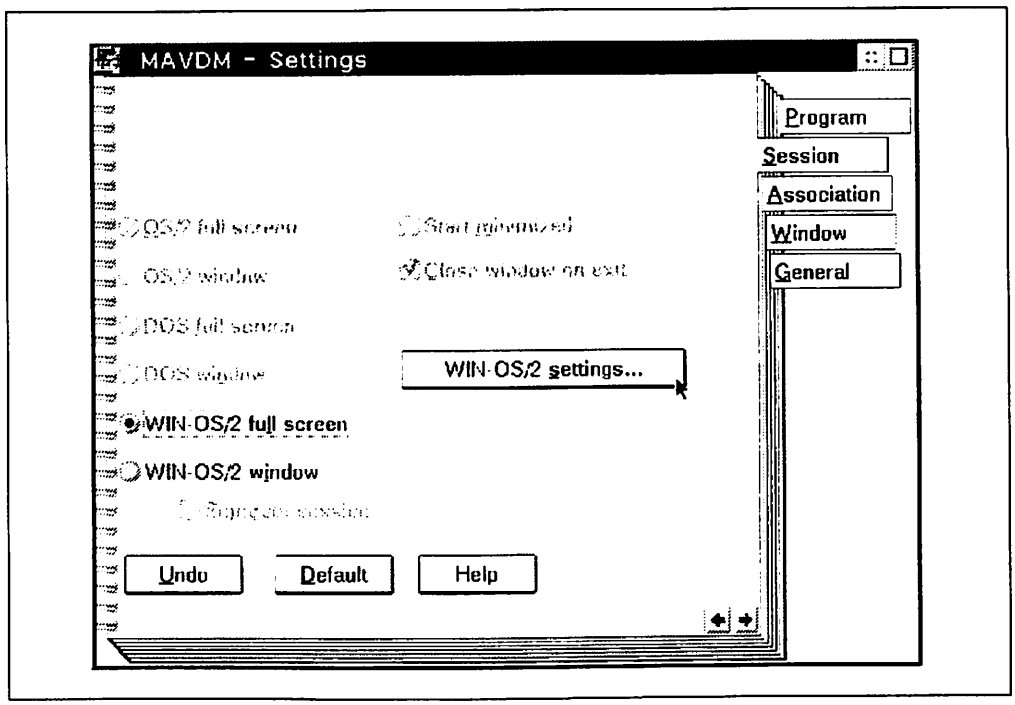


Figure 75. WIN-OS/2 3.1 - Defining a Full Screen Session

The second type is the WIN-OS/2 windowed session, also called the **WIN-OS/2 seamless** session. This is selected by clicking the left mouse button on the radio button next to the **WIN-OS/2 window** option, as shown in Figure 76 on page 102.

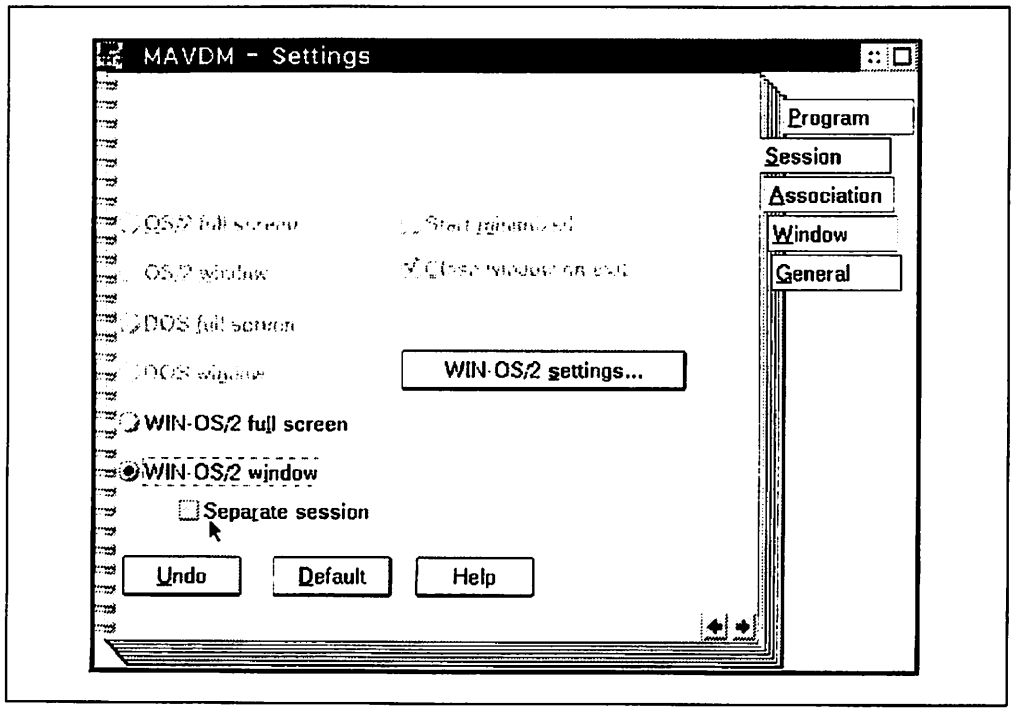


Figure 76. WIN-OS/2 3.1 - Defining a Common Seamless Session

Once the WIN-OS/2 window option is selected, the **Separate session** option below it will be darkened, which means that this option is now selectable. By default, **Separate session** is not selected and thus it is a definition for WIN-OS/2 Common seamless session.

The third type is the WIN-OS/2 windowed separate session, also called the WIN-OS/2 separate seamless session. This is set up by selecting the **Separate session** option as shown in Figure 77 on page 103. Of course, the **WIN-OS/2 window** session option must first have been selected.

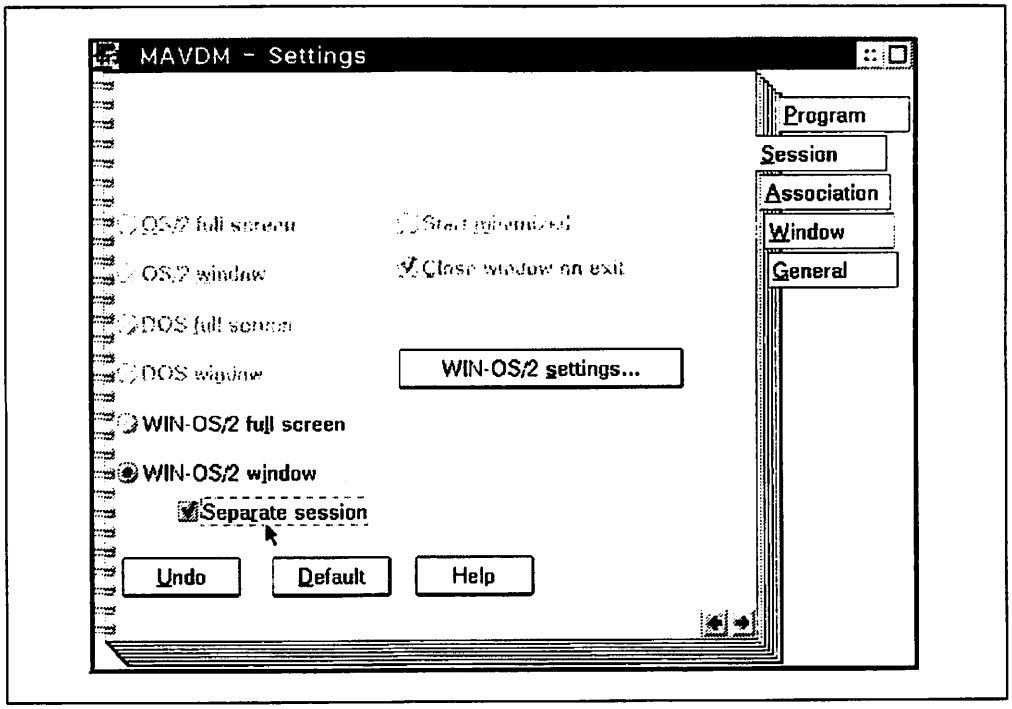
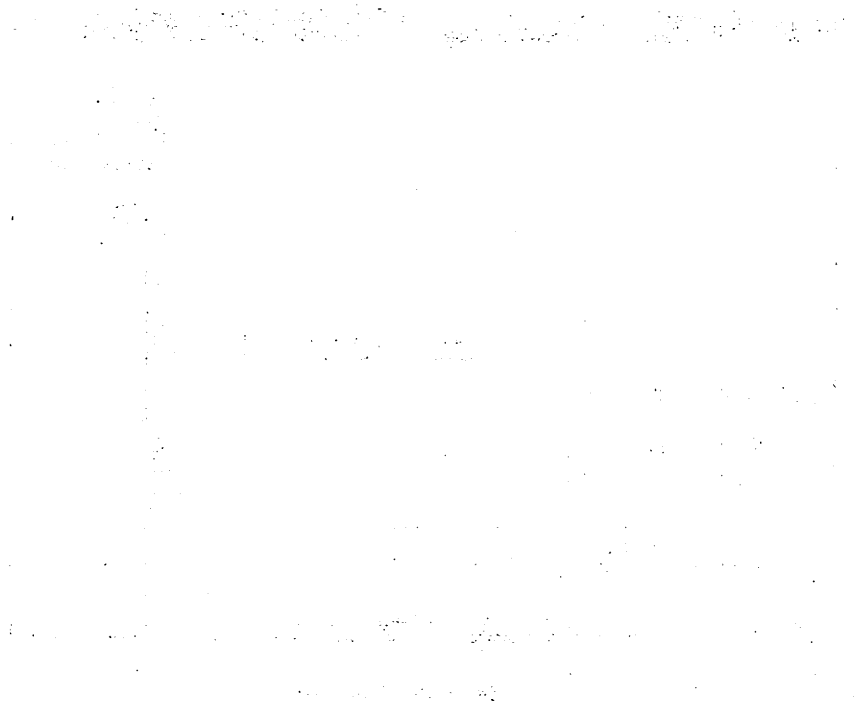


Figure 77. WIN-OS/2 3.1 - Defining a Separate Seamless Session

Note

To reduce system resource usage, use common seamless sessions whenever possible, since only one VDM will be started for all the WIN-OS/2 seamless sessions.



Chapter 8. Presentation Manager Enhancements

The intent of this chapter is to highlight the enhancements and changes that have been incorporated into the Workplace Shell (WPS) and Presentation Manager (PM). Some of the changes that have been made to WPS and PM are:

- A new 32-bit graphics engine
- SVGA support for leading adapters
- 32-bit seamless device driver support for XGA, XGA-2, SVGA, 8514/A, and VGA
- New ISO fonts that conform to the ISO standard 9241-3
- New Palette Manager Support

Many of these changes apply to preloaded OS/2 2.00.1 and OS/2 2.0 with Service Pak XR06055 applied, as well as to OS/2 2.1. For exact details, see Appendix A, "OS/2 2.00.1 and Service Pak XR06055 Enhancements" on page 289.

The overall operational aspects of Presentation Manager and the Workplace Shell have not changed much since OS/2 2.0 was first released, but many of the internal operating system modules have been revised or rewritten. The major enhancement that has been made to Presentation Manager is the introduction of the new 32-bit graphics engine and the associated 32-bit seamless display drivers. This new graphics engine enables 32-bit applications to fully exploit the OS/2 graphics environment, without recompilation.

8.1 Overview of Presentation Manager

Presentation Manager (PM) is the Graphical User Interface (GUI) of OS/2. It is responsible for displaying windows and other information on the screen and for handling input from the keyboard and pointing devices. Its objective is to make the system simple and intuitive to use.

As well as being the basis of the OS/2 GUI, PM also provides an interface for use by programmers in developing applications and presentation drivers. The structure of PM from the perspective of the application programmer is shown in Figure 78 on page 106.

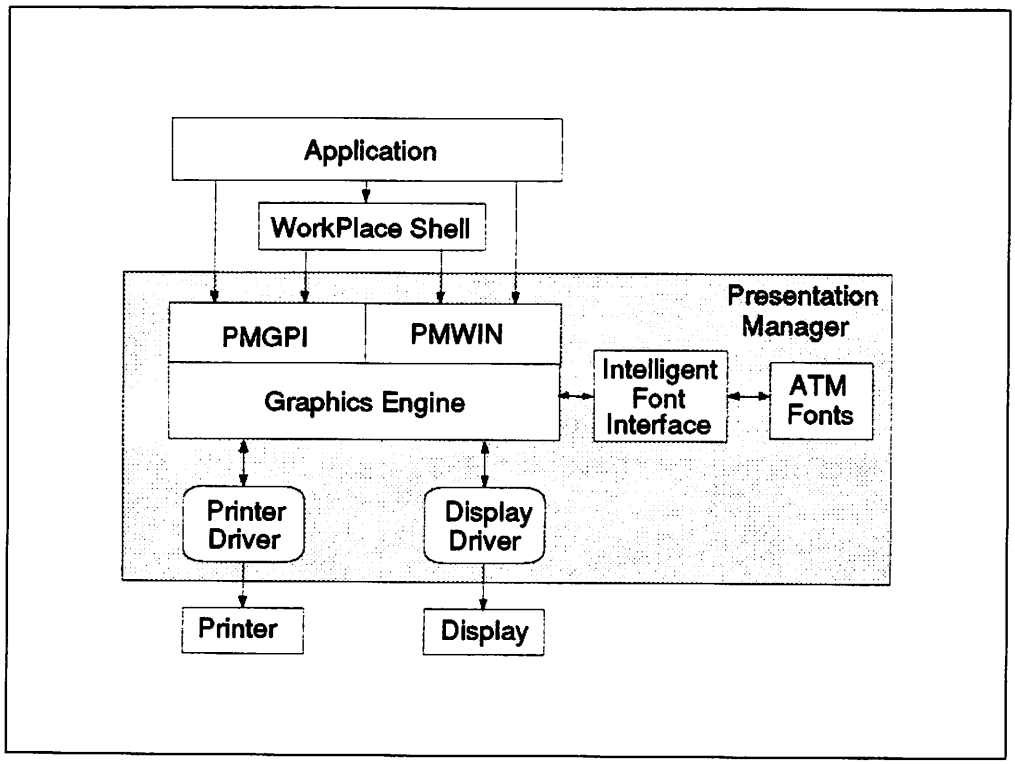


Figure 78. Structure of Presentation Manager

Presentation Manager consists of four main components:

PMGPI The Graphics Programming Interface provides the means for an application to use the graphical environment. For example, to draw a line or circle or to write text at a given position on the screen, a call is made to PMGPI.

PMWIN The Window Manager is responsible for creating, maintaining and destroying windows on the PM desktop. For example, PMWIN provides the mechanism to pop-up a dialog or message box from an application.

PMGRE The Graphics Engine is at the heart of the PM system. It is not called by applications directly, but is used by PMGPI and PMWIN on behalf of the applications. It works closely with the display and printer drivers.

PMDDs The PM Device Drivers (display drivers and printer drivers) are the part of the system that cause information to be displayed on the screen or printer. PMDDs are normally specific to one device. For example, a VGA display and an XGA display require different display drivers because of the different capabilities of those devices.

8.2 32-bit Graphics Engine (PMGRE)

The new 32-bit graphics engine has been designed to give faster and more accurate graphics performance. It can handle larger, more complex graphics than was previously possible. Because the 32-bit engine is a completely new implementation written in C rather than assembler, many of the problems associated with the old engine have been eliminated. Maintenance becomes easier and the engine could easily be ported to another architecture.

8.2.1 Function of the Graphics Engine

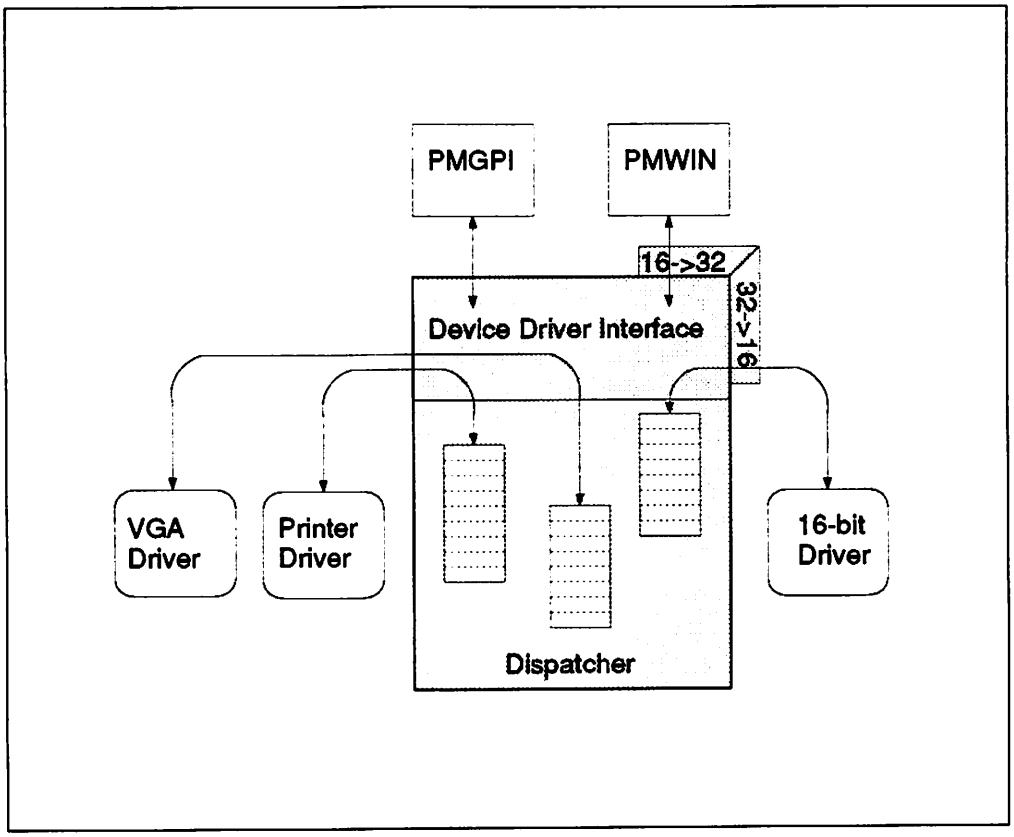


Figure 79. Structure of the 32-Bit PM Graphics Engine

The structure of the 32-bit PM Graphics Engine is shown in Figure 79. The graphics engine consists of two parts - the **Device Driver Interface** (DDI) and the **Dispatcher**.

The DDI is called by the other parts of PM (PMGPI and PMWIN) when they require the engine to perform a function. The DDI can also be called by the drivers.

The Dispatcher contains a set of Dispatch Tables (one per driver). Each Dispatch Table has entries for each of the possible functions that the engine may be called upon to perform. When a driver is first loaded, usually when the system is booted, the engine queries the driver to see what its capabilities are. For example, some drivers may be capable of drawing circles, others may not. These capabilities are written into the Dispatch Table for that device.

There is a minimum set of basic functions that a driver must support. All other functions can be provided by the engine if necessary. A driver will generally support more advanced functions if it is capable of doing it better or faster than the engine. For example, some drivers support complex clipping areas, others do not.

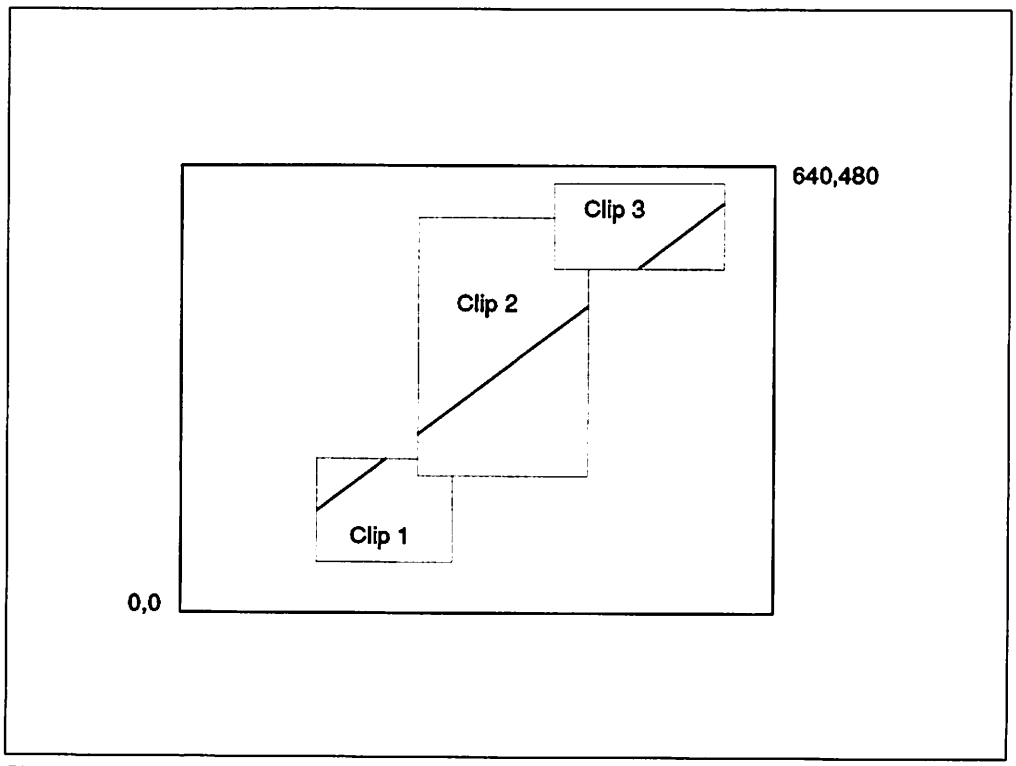


Figure 80. Example of Clipping

Figure 80 shows the output required by an application that draws a straight line from the bottom left of the screen (0,0) to the top right (640,480), having already set up three clipping regions. The engine will first query the dispatch table to see if clipping is supported by this driver.

If clipping is supported, the engine will tell the driver to draw the line and clip it to the three regions. The hardware itself may support clipping, for example if it is an XGA type of adapter. In this case, the driver passes the line information to the hardware and lets it do the clipping. If the hardware does not support this function, the driver has to use a clipping algorithm to break the line down into its three component parts. The three lines are then passed to the hardware and are drawn individually.

If clipping is not supported by the driver, the engine will perform the clipping algorithm and then instruct the driver to draw three separate lines on the screen.

Note the 16/32 bit conversion layer in Figure 79 on page 107. Because the engine is now implemented as 32-bit code, calls from 16-bit code need to pass through a **thunk**. The thunk converts the 16-bit call into a 32-bit call. A similar process is used when the engine calls a 16-bit driver. The 32-bit engine call passes through a thunk before it can be processed by the 16-bit driver.

8.2.2 Benefits of a 32-bit Graphics Engine

The 32-bit graphics engine has been provided in order to support 32-bit drivers. The 16-bit drivers also continue to be fully supported by the new 32-bit engine. Drivers using the 32-bit flat memory model are less complicated to develop than those using the 16-bit model. They are potentially more efficient and therefore faster than their 16-bit equivalents because of the lack of segmentation in the 32-bit memory model.

All API calls were previously limited by the 64KB segments available in a 16-bit implementation. By providing a 32-bit engine, it has also been possible to increase or remove many of the limits in the graphics model. The removal of these limits should make things much easier for application programmers. Users will also benefit because their applications can have more capabilities than were previously possible.

8.2.3 Increased Limits

The 16-bit segmented architecture of the earlier graphics engine led to some restrictive limits. For example, the number of bitmap handles and points on a polyline were limited. Table 20 and Table 21 shows some of the limits that have been greatly increased through the introduction of the 32-bit architecture.

Table 20. Comparison of Limits of 16-Bit and 32-Bit Graphics Engine

Resource type	**16-bit**	**32-bit**
Points in a polyline	64K	128M
GPI paths	1400 points	8M points

Table 21. Further 32-Bit Graphics Engine Limits

Resource type	**32-bit**
PS / DC handles	16K
Bitmap handles	64K
Region handles	64K
Font handles	64K

Additionally, the shared resource limits for resource types such as bitmaps and fonts, are approximately equal to the system limits due to the flat memory model. There is no longer the restriction of 8000 shared resources imposed by the number of LDT (Local Descriptor Table) entries per process.

8.2.4 New GpiPolygons API

This API is provided for the high speed drawing of polygons. It was available to applications in OS/2 2.0, in a 16-bit implementation. The new 32-bit implementation removes the restrictions and provides a faster, more powerful mechanism for drawing one or more polygons.

For more information about the GpiPolygons API, please see the *OS/2 Version 2.0 - Volume 3: Presentation Manager and Workplace Shell.*

8.2.5 Transparency Color Mapping

Transparency Color Mapping is a performance function which allows one application to "paste" an image or graphic over another.

It can be used by application programmers in the following way.

A new background mix option, BM_SRCTRANSPARENT, has been added, for the GpiBitBlt and GpiDrawBits API calls. When this option is used, pels from the source bitmap which match the presentation space background color are not be

copied to the output bitmap, effectively leaving those pels in the output unchanged. This provides for a transparent overlay function.

A second new background mix option, BM_DESTTRANSPARENT, has also been defined for bitmap GpiBitBlt and GpiDrawBits API calls. If this option is used, then pels from the source bitmap will only be copied to destination pels that match the presentation space background color. This provides for a transparent underlay function.

8.2.6 PEL Translation

PEL translation is used for high speed color changes, shading, gaussing or brightening images.

This function is also used by application programmers.

On the Drawbits API, the application can provide a palette mapping instead of a color table. This palette mapping will be used to define the colors in the bitmap. This is much quicker then the color table approach, because the driver does not have to translate all of the bitmap bits from the provided color table to the real color table.

8.3 32-Bit Seamless Display Drivers

With the introduction of the 32-bit graphics engine, 32-bit display drivers are now supported. The 16-bit drivers also remain fully supported.

8.3.1 Function of a Display Driver

As discussed in 8.2.1, "Function of the Graphics Engine" on page 107, the display driver and the engine work very closely together. Display drivers are written to access all of the features of the display adapter in the most efficient way possible. Any functions that cannot be handled by the driver are handled by the engine. The engine segments a complex request into a series of less complex ones that the driver can handle.

8.3.2 Summary of Changes

OS/2 2.0 supports 16-bit display drivers for the following display adapters:

- CGA
- EGA
- VGA
- 8514/A
- XGA

OS/2 2.1 provides 32-bit display drivers for VGA, 8514, SVGA and XGA display adapters.

The 16-bit CGA and EGA display drivers also continue to work in conjunction with the new 32-bit graphics engine in OS/2 2.1.

OS/2 2.0 supported seamless WIN-OS/2 sessions when the VGA display driver was being used. With OS/2 2.1, it is now possible to run seamless WIN-OS/2 sessions with VGA, 8514, SVGA and XGA adapters.

Many of these 32-bit seamless display drivers were also shipped with preloaded OS/2 2.00.1 and Service Pak XR06055 for OS/2 2.0. For exact details, see Appendix A, "OS/2 2.00.1 and Service Pak XR06055 Enhancements" on page 289.

8.3.3 SVGA Support

Super Video Graphic Adapters (SVGA) have become very popular in the marketplace. There are a number of manufacturers of SVGA chip sets, as well as a number of variations on how these have then been implemented as SVGA adapters. IBM has also recently implemented SVGA adapters with some of its new PCs, such as the PS/ValuePoint range.

IBM has been working very closely with these SVGA manufacturers to develop a generic SVGA driver that can be used in OS/2 2.0. Section 4.9.2, "Video System Support in OS/2 2.1" on page 54 contains a list of SVGA adapters that are supported under OS/2 2.1.

8.3.4 XGA and XGA-2 Support

The XGA display drivers support both XGA and XGA-2 display adapters. Some functions, such as DMQS, are only available to XGA-2 display adapters.

The advantages of XGA-2 over XGA are:

- 16-bit color support, giving 64K colors
- DMQS support
- Improved performance
- Non-interlaced capability (if a non-interlaced display is attached)

8.3.5 DMQS and XGA-2

Display Mode Query and Set (DMQS) enables the display adapter to find out the capabilities of the attached display, such as screen resolution, and to set the display configuration. DMQS has been implemented for the XGA-2 adapter, and is supported in the XGA display driver.

DMQS has been introduced so that drivers written for the XGA Subsystem will work regardless of what display is attached.

The XGA subsystem has a programmable frequency generator. This means that many different frequencies can be generated by the same display adapter. It is the display attached to the subsystem that determines the number of video modes available to the user, rather than the display adapter as was previously the case.

DMQS is needed so that drivers and applications can determine screen size, ISO font compliance, and also screen characteristics, such as Color/Mono or LCD/CRT.

There are a number of benefits that result from DMQS. Display driver updates can be released independently of display or adapter releases. It also means that the XGA subsystem is not released with specific modes, rather the mode is dependent on the display. Most importantly, many more displays, including non-IBM displays, can be attached to the XGA subsystem.

The XGA display driver uses the primary DMQS data to determine the slot numbers of all XGA adapter instances. DMQS primary data also includes the monitor ID for the display attached to each XGA-2.

Once the monitor ID is known, the corresponding DGS file is read in and stored. The DGS files are stored in the directory \XGA$DMQS, and contain hardware information about the the display capabilities and also has the information needed to properly set the display into high resolution mode.

Some OEM displays return the 8514 monitor ID. DMQS Override has been introduced to enable this monitor ID to be overriden; this is implemented through the file \XGA$DMQS\XGASETUP.PRO which tells the display driver the real monitor ID to use.

Without DMQS override support, the driver would treat these displays as an 8514 even though they may have advanced capabilities.

To use DMQS Override, access the System icon within the System Setup folder. The second page of the screen part of the notebook provides a utility to set up DMQS Override. For more details, refer to 3.3, "Display Driver Install Program" on page 30.

To cancel the effect of DMQS Override, erase the file \XGA$DMQS\XGASETUP.PRO.

For additional information on DMQS and the operational aspects of XGA, please refer to the bulletin *IBM PS/2 and PS/ValuePoint Subsystems, GG24-4002.* This document also contains information on how to program the XGA adapters.

8.4 Palette Manager

The Palette Manager has been added to Presentation Manager to allow applications greater flexibility and to provide better-looking graphics. It can be used in configurations where the hardware supports at least 256 colors on the screen at the same time, such as XGA, SVGA or 8514 adapters.

An application can now define some or all of those 256 colors for its own use. Previously, applications had to make do with a "closest fit" color or a "dithered" color.

Presentation Manager applications can take advantage of this feature. The Palette Manager implementation is also compatible with Windows applications. So an application written to take advantage of the Windows Palette Manager will also benefit from the Presentation Manager Palette Manager.

Because the Palette Manager function requires access to the video hardware at a low level, it is implemented through a combination of the display drivers and the graphics engine. The API was available in OS/2 2.0, but the function has only been available with the 32-bit graphics engine and associated 32-bit display drivers.

The PM Palette Manager interoperates with the WIN-OS/2 Palette Manager, in order to coordinate the color palettes between the two environments. Palette changes may still be seen when switching between PM and WIN-OS/2 applications, just as they can happen switching between two PM applications.

8.4.1 Colors in Presentation Manager

A brief description of the use of colors in Presentation Manager follows. This topic is discussed more fully in Chapter 7 of the *OS/2 2.0 Programming Guide Volume 3*.

Colors are defined in terms of their red, green and blue components (RGB). The colors which the hardware is capable of displaying at any given time are stored in the **physical color table**. On a VGA system, the physical color table has 16 RGB entries, and on an SVGA, XGA or 8514 system it has 256 RGB entries or more. There is only one physical color table in the system and it is shared by all applications.

When an application creates a Presentation Space (PS), a default **logical color table** is associated with that PS. The logical color table can be of variable length and is typically smaller than the physical color table. The application can change the RGB values in the logical color table and can access the colors by index rather than by RGB if desired.

When an application specifies a color, it does so through its logical color table. The color specified in the logical table is compared with the entries in the physical table. If a match is found, that color is displayed on the screen. If it does not match, the closest available color in the physical table is used. If the color will be used to fill an area, *dithering* may be used by the system to match more closely with the color specified in the logical table.

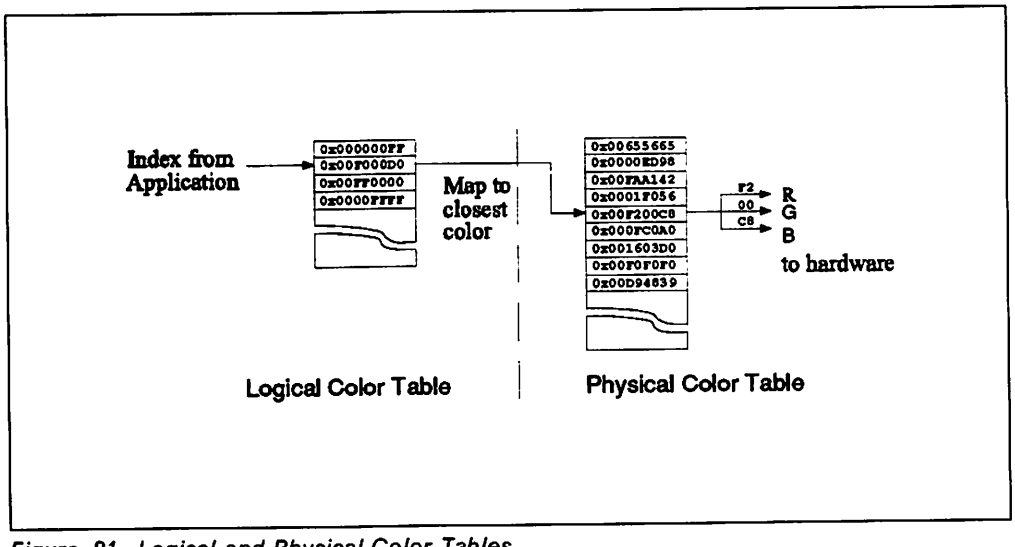


Figure 81. Logical and Physical Color Tables

Figure 81 shows the mapping of an index supplied by an application, through the logical color table and the physical color table and onto an RGB value that the hardware can use to display the color.

Palette Manager provides another way for an application to specify its colors. It should generally be used only in situations where the color table approach is not powerful enough. The use of palette management functions can have an effect on the output being displayed by other applications in the system. The color table and Palette Manager methods should not be mixed within an application.

The system contains one **physical palette**. It is equivalent to the physical color table described in Figure 81. An application can then set up its own palette with

specific colors and replace part (or all) of the physical palette with its own palette.

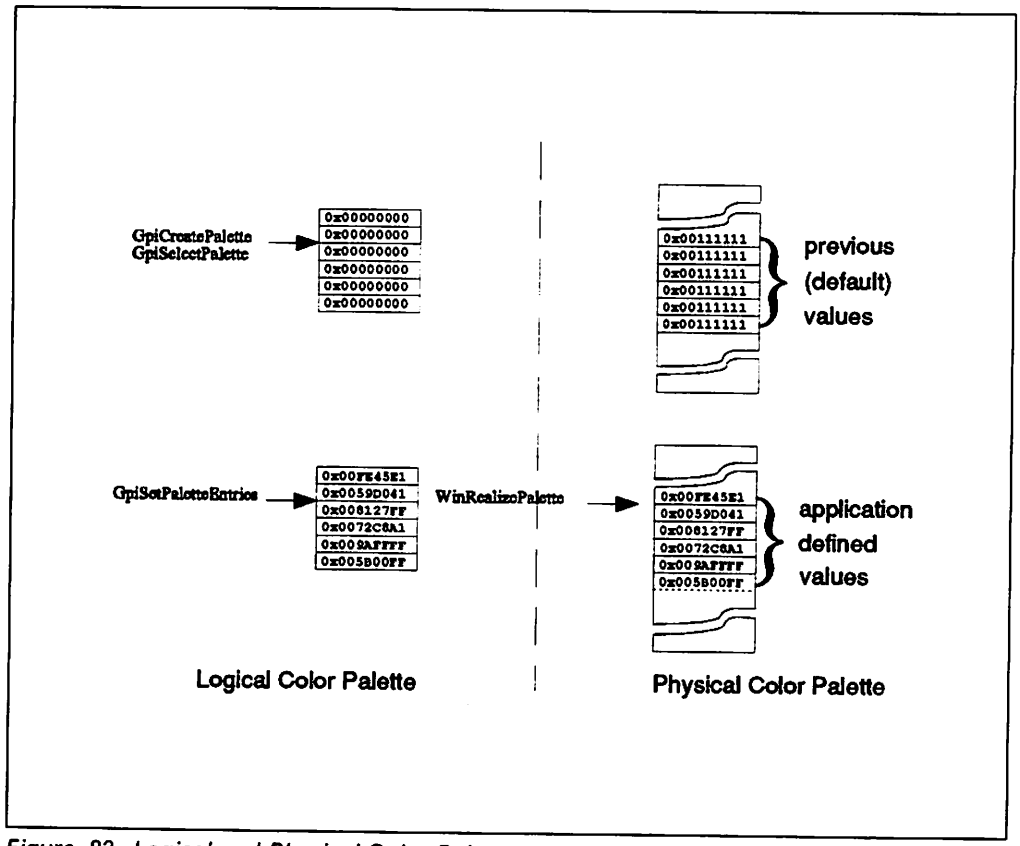


Figure 82. Logical and Physical Color Palettes

Figure 82 shows how an application creates a palette using the **GpiCreatePalette** function and selects it for the Presentation Space using **GpiSelectPalette**. It can also change color values in the palette using **GpiSetPaletteEntries**, and replace part of the physical palette with the new values using **WinRealizePalette**.

An application can specify the size of its own palette (up to the size of the physical palette). When more than one application uses the Palette Manager functions at the same time, there may not be enough entries in the system color palette to support all applications.

For example, if the physical palette has 256 entries and two applications each create and realize palettes of 100 entries, each can use the full range of colors requested. However, if a third application creates a palette of 100 entries, there are not enough entries available in the physical palette to satisfy all of the applications. The following is the outcome:

1. The application in the active window is able to use all colors in its palette.
2. The palette of the most recently used application is considered next. If there are enough entries available in the physical palette, some (or all) of them are used by this application. If not enough entries are available, not all of the application's colors will be placed in the physical palette.
3. The other applications are each considered in turn (most recently used to least recently used). Any remaining entries in the physical palette are allocated.

4. If all palettes from some of the applications have *not* been satisfied in the physical palette, the system sends a message, WM_REALIZEPALETTE, to all applications running on the PM desktop, except the active one.
5. Any application using palettes should process the the WM_REALIZEPALETTE message. They should call WinRealizePalette and then invalidate their windows so that they get repainted. Palettes of applications that are inactive do not get realized in the same way as applications that are active if there are not enough free entries in the physical color palette to support all applications. The palettes of the inactive applications may get mapped onto the closest available colors. This can lead to some unexpected results while an application is running in the background.
6. When an inactive application is made active, it should realize its palette again. When it repaints its window, its output will be correct again.

8.4.2 New Palette Manager APIs

Several new APIs were added to PM in Version 2.0. Because there were no display drivers that were capable of handling these functions, applications were unable to take advantage of them. With the introduction of the Palette Manager APIs for the XGA, SVGA and 8514 drivers, these PM APIs can now be used. One of the APIs is part of PMWIN, the others are all in PMGPI.

Table 22. Palette Manager Functions

Function Name	Description
GpiAnimatePalette	Changes the color values of animating indexes in a palette
GpiCreatePalette	Creates and initializes a color palette
GpiDeletePalette	Deletes a color palette
GpiQueryPalette	Determines the palette currently selected in a presentation space
GpiQueryPaletteInfo	Determines the RGB, the index values, or both, for the currently loaded colors
GpiSelectPalette	Selects a new palette for the presentation space
GpiSetPaletteEntries	Changes entries in a logical palette
WinRealizePalette	Maps a logical palette into the system palette

Table 22 lists the Palette Manager functions. These functions are also described in the *OS/2 Programming Guide, Volume 3.*

Before using the Palette Manager APIs, an application should first query whether the Palette Manager is supported in the current system. This query is achieved through a call to **DevQueryCaps** with the **CAPS_ADDITIONAL_GRAPHICS** option.

8.4.3 Display Adapters Supporting Palette Manager

Palette Manager can only be used in systems that can display at least 256 colors at the same time. The XGA, SVGA and 8514 drivers support the Palette Manager functions.

8.5 ISO Standards for Fonts and Displays

The International Organization for Standardization (ISO) has developed several standards that fonts and displays can conform to. To conform to the ISO 9241-3 standard the displays must be designed to:

- Reduce the amount of flicker
- Display sharp, crisp characters
- Increase brightness
- Provide stable images
- Reduce glare
- Provide high contrast

These standards also look at various aspects of how fonts are displayed. For instance:

- The *internal contrast ratio* of characters must be greater than a specified minimum value. The internal contrast ratio is a measure of how clearly the foreground of a character stands out against its background. This is influenced by the width of the lines within a character and the separation of those lines. In order to meet the internal contrast requirement, there is a minimum size that a font can be displayed at and still pass the ISO 9241-3 test.
- The widths of lines within a character should be consistent. For example, in the letter **H**, the vertical and horizontal strokes should be the same thickness.
- All characters within a font must be distinct from each other. For example, the number "0" and the capital letter "O" must be different.
- Descenders, ascenders and accents must not collide with each other. When characters are underlined, they must retain their clarity.

In order to meet these ISO 9241-3 standards, some of the standard fonts have been modified. In general, the effect is for character widths to increase slightly. The affected fonts are:

- System Proportional
- Courier
- Helv
- TmsRmn

8.5.1 New ISO Compliant Fonts

The new ISO System Proportional font replaces the old one. All other ISO 9241-3 conforming fonts are an addition to the previous fonts and do not replace them. The system does not force you or an application to choose these fonts. In some situations, it may not be appropriate to choose one of the new fonts because the increased character widths may prevent text from being displayed in a place where there was previously no problem.

By default, the Workplace Shell and VIO sessions (OS/2 and DOS full screen and windowed sessions) use the System Proportional ISO font.

The new fonts have the same name as the old ones with the addition of "ISO" at the end of the name. So, the new fonts are:

- Courier ISO
- Helv ISO
- TmsRmn ISO

8.5.2 Non-ISO Warning Messages

On a hardware system that supports the ISO 9241-3 standard (non-interlaced display adapter and display), you will be warned if a non-ISO font has been chosen in any of the following places:

- Standard font dialog - this is a dialog box which can be used by any PM application to allow the user to choose a font.
- Workplace Shell **Font Palette** and from the **Edit Font** dialog that is displayed when the **Change Font** button is selected on notebook controls. These dialogs are similar to the standard font dialog but are modified slightly by the Workplace Shell.

If you choose a non-ISO font a warning message is displayed at the bottom of the font dialogs listed above. The system does not force you to choose an ISO 9241-3 font. A warning message will only be displayed if the font you have chosen has been tested for ISO 9241-3 compliance and has been found to fail. A font which has not been tested will never cause a warning message to be displayed.

Chapter 9. Workplace Shell Enhancements

Some changes have been made to the Workplace Shell in OS/2 2.1, in order to improve both the usability and the visual appearance.

In addition, the INI file handling has been rewritten in order to improve speed and usability. Much of the configuration information for the Workplace Shell, such as icon settings and position, is stored in the two files OS2.INI and OS2SYS.INI.

Finally, the name of the desktop directory has been changed to **Desktop** for new OS/2 2.1 installations on both FAT and HPFS file systems.

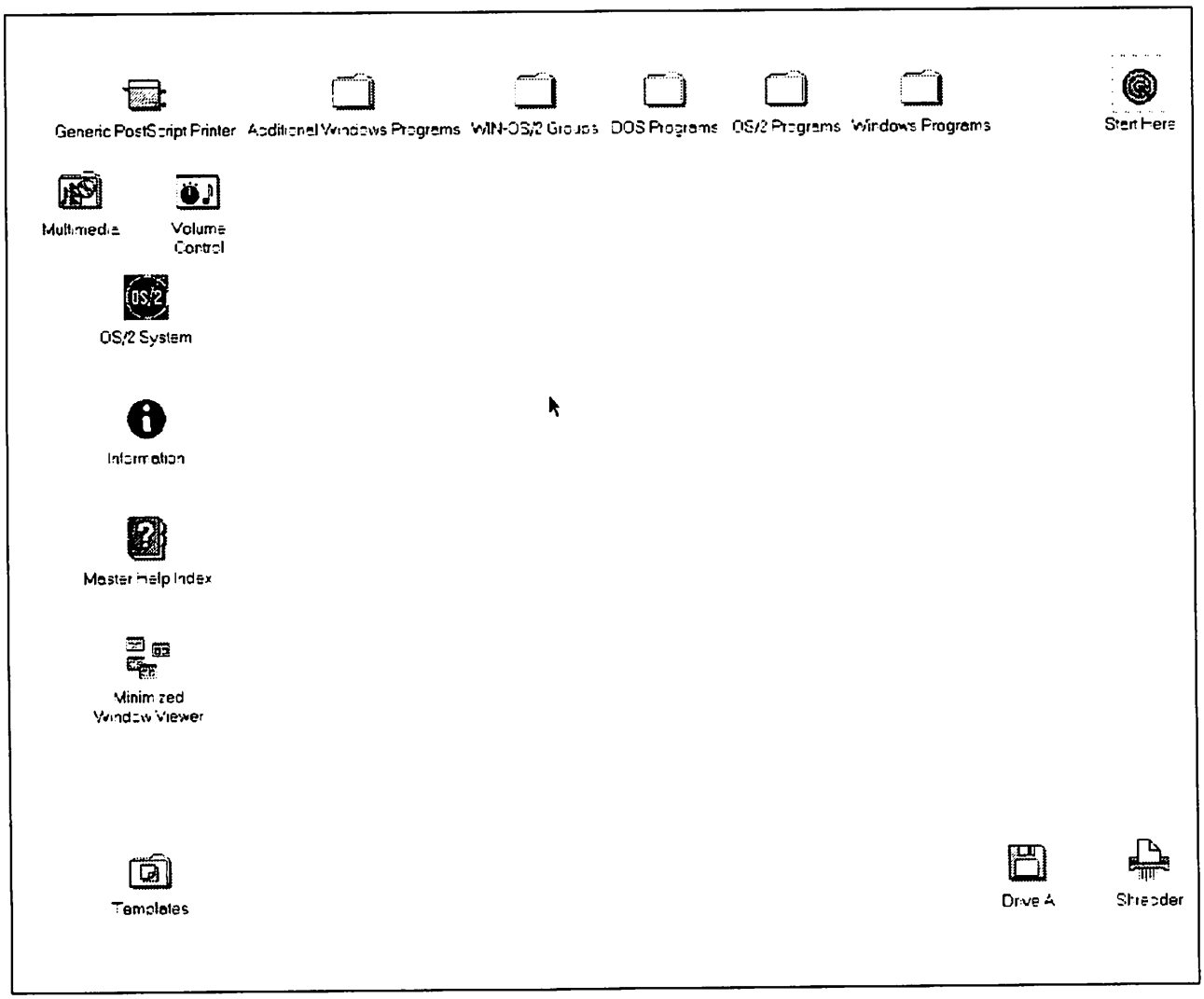


Figure 83. Workplace Shell - OS/2 2.1 Desktop

A typical initial appearance of the Workplace Shell desktop is now as shown in Figure 83. This is at XGA resolution, and includes MMPM/2 1.1 as well as migrated DOS, Windows and OS/2 programs.

9.1 Visual Enhancements

A number of small but useful visual enhancements have been made to the Workplace Shell.

A **System Setup** entry has been added to the desktop pop-up menu, as shown in Figure 84.

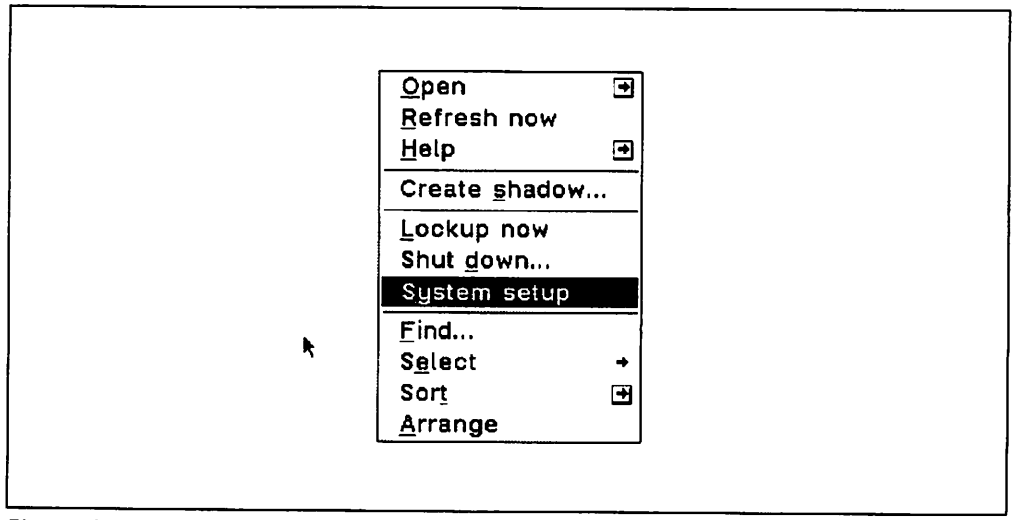


Figure 84. Workplace Shell - New Desktop Pop-Up Menu

Choosing the System Setup entry will open the System Setup folder, as shown in Figure 85. This provides a fast way to access commonly used setup utilities for OS/2 2.1.

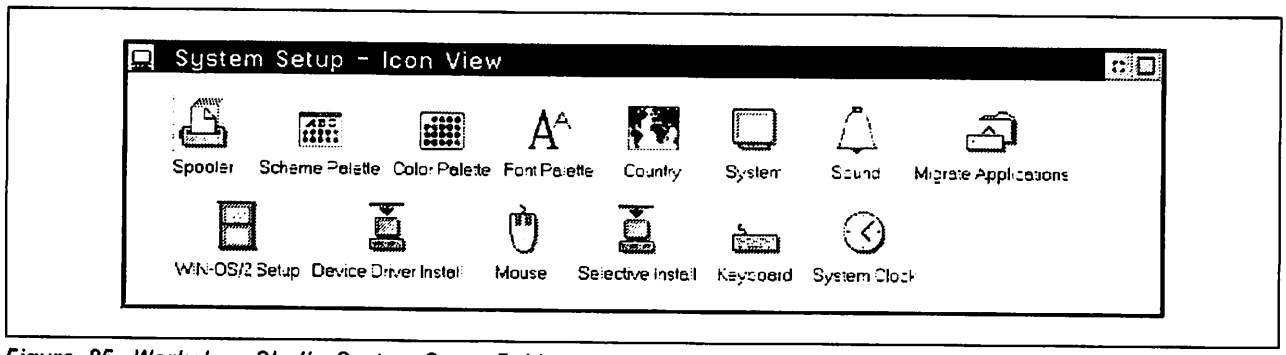


Figure 85. Workplace Shell - System Setup Folder

As shown in Figure 83 on page 119, the standard folder icon is now used for the OS/2, Windows and DOS program groups, instead of a program group icon. This provides a more natural visual integration of program objects with other objects in the Workplace Shell.

The icon for CD-ROM drives, as shown in Figure 86 on page 121, is now always used even if no data CD-ROM is present in the CD drive at boot time.

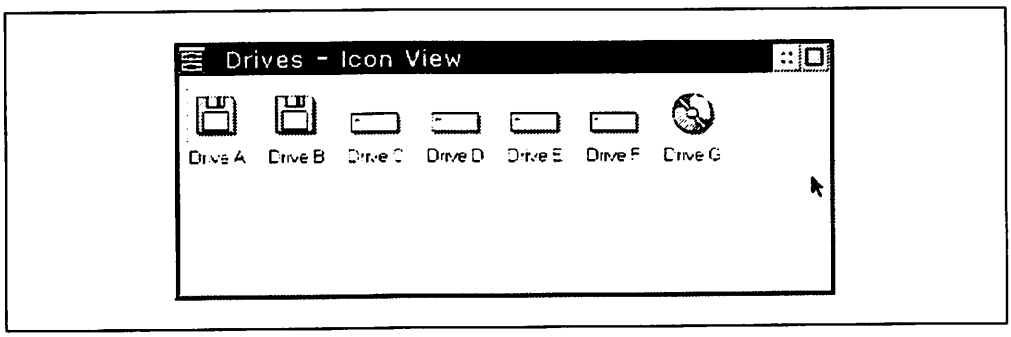


Figure 86. Workplace Shell - CD-ROM Drive Icon

The pop-up menu for CD-ROM drives now contains options for software-driven eject, as well as software lock and unlock. This is shown in Figure 87. These options are also available on the pop-up menus for diskette drives and optical drives which provide hardware support for these features.

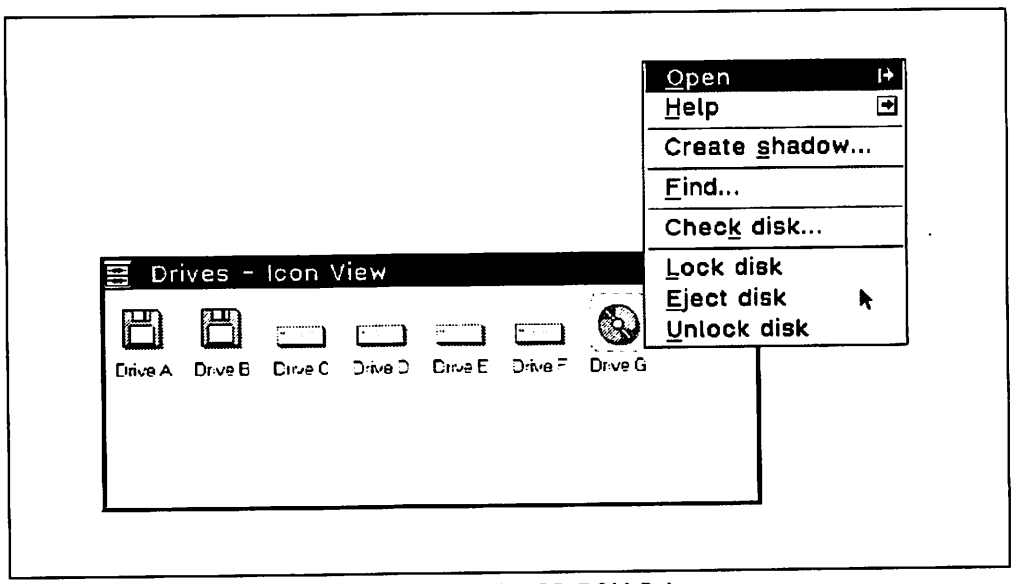


Figure 87. Workplace Shell - Pop-up Menu for CD-ROM Drive

Perhaps the most striking change to the appearance of OS/2 2.1 is that the notebook control has been changed to have a spiral binder, just like a real notebook. This is shown in Figure 88 on page 122.

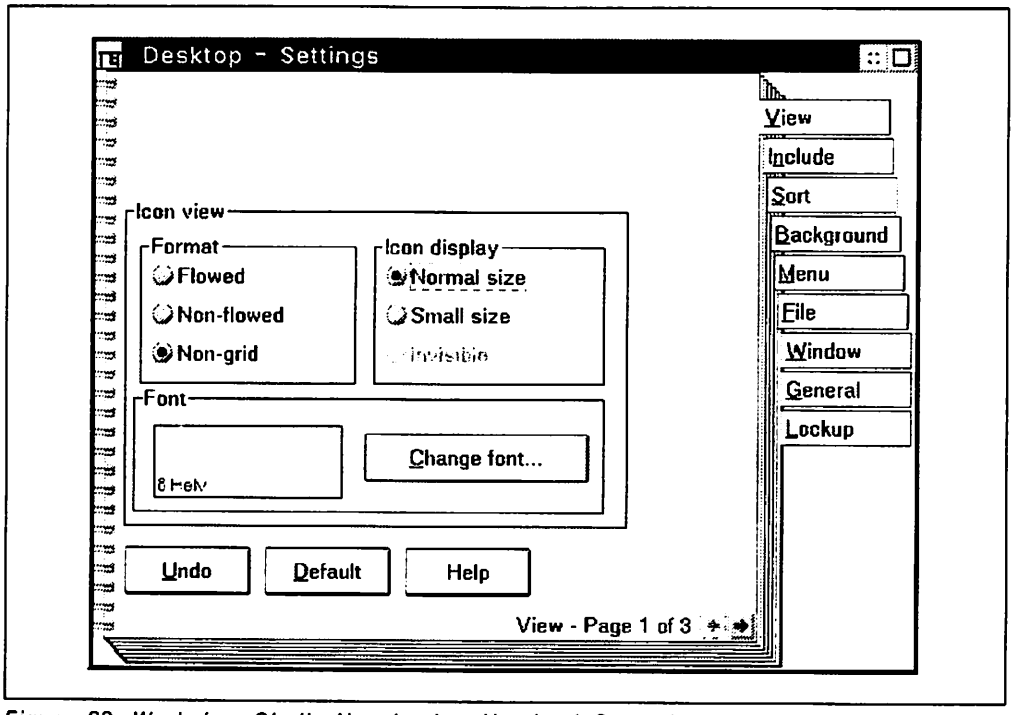


Figure 88. Workplace Shell - New Look to Notebook Control

9.2 Settings Notebook Drag/Drop Enhancements

Two drag/drop enhancements have been added for use with the Settings notebook of Workplace Shell objects:

- Adding program objects to a menu (on the Menu page)
- Changing the object icon (on the General page)

9.2.1 Adding Program Objects to a Menu

Program objects can now be easily added to the pop-up menu of folders and other objects that provide a Menu page in the Settings notebook. A typical use of this would be to add a program or command prompt to the desktop pop-up menu.

Open the **Settings** notebook of a folder, and select the **Menu** page. Then simply drag the icon of a program object and drop it on the **Actions on menu** list box.

An example of this is shown in Figure 89 on page 123 and Figure 90 on page 123 for the case of adding the MMPM/2 Compact Disc player to the desktop menu.

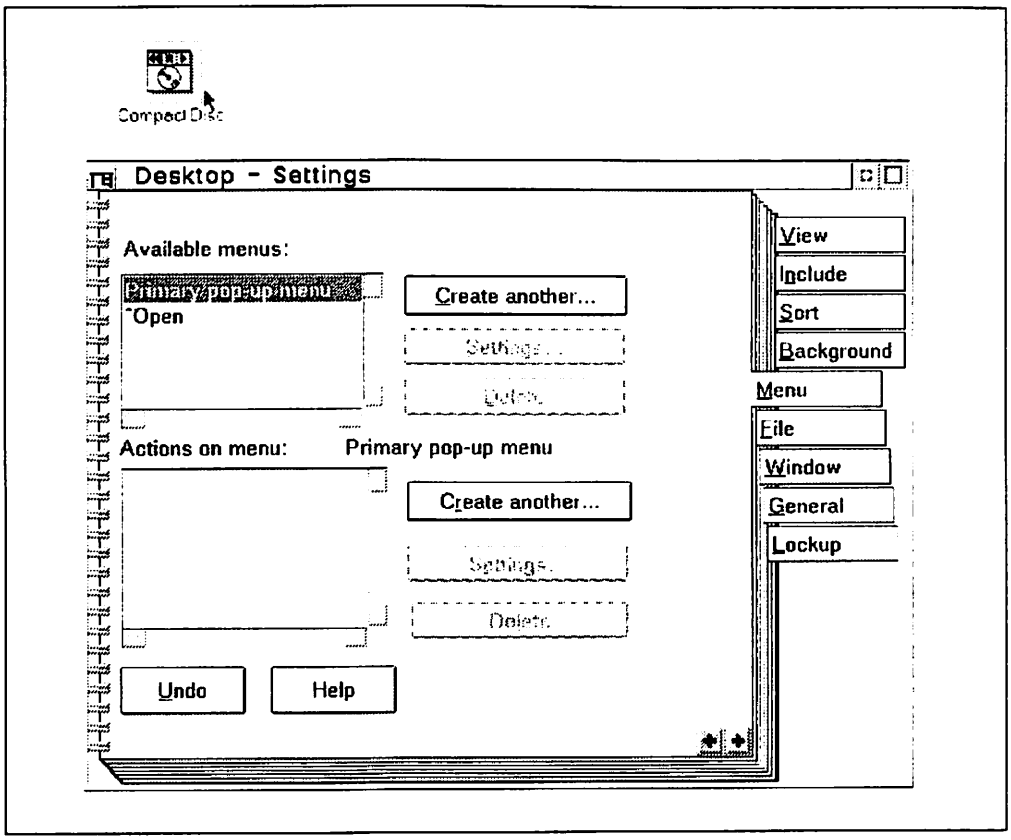


Figure 89. Workplace Shell - Dragging a Program Object to the Menu Page

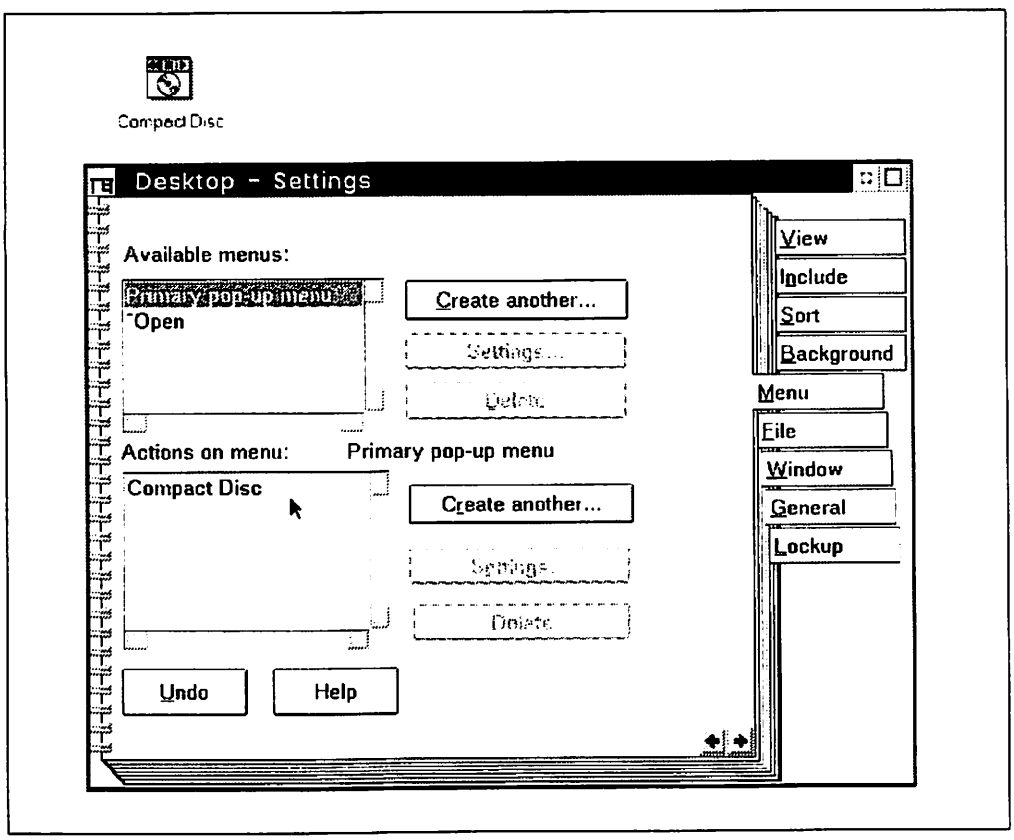


Figure 90. Workplace Shell - Adding a Program Object to the Menu by Drag/Drop

Once the program object has been added to the menu, it is immediately available, as shown in Figure 91 on page 124.

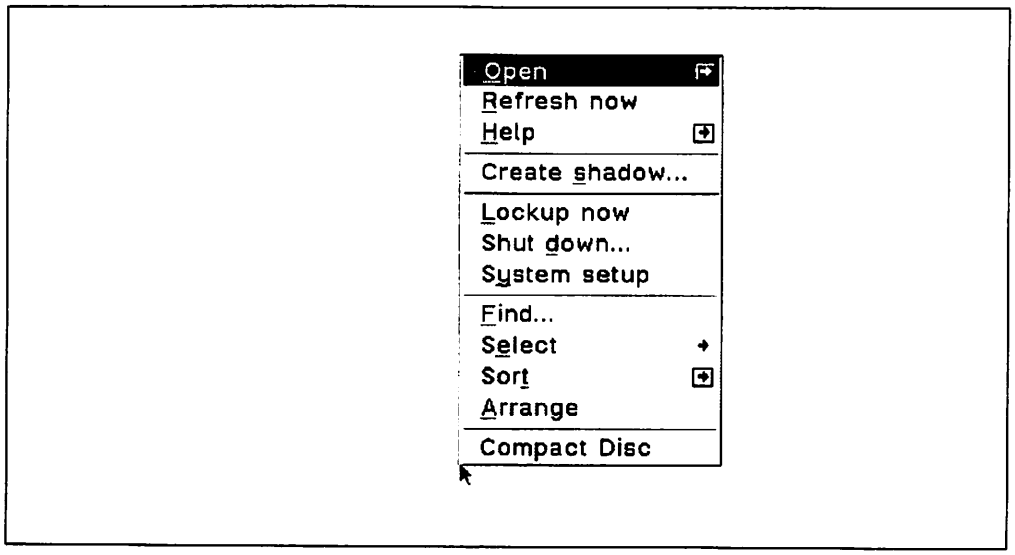


Figure 91. Workplace Shell - Desktop Pop-Up Menu with Program Object Added

9.2.2 Changing the Icon for an Object

The icon for any Workplace Shell object can now be easily changed by dragging a new icon onto the General page of the Settings notebook.

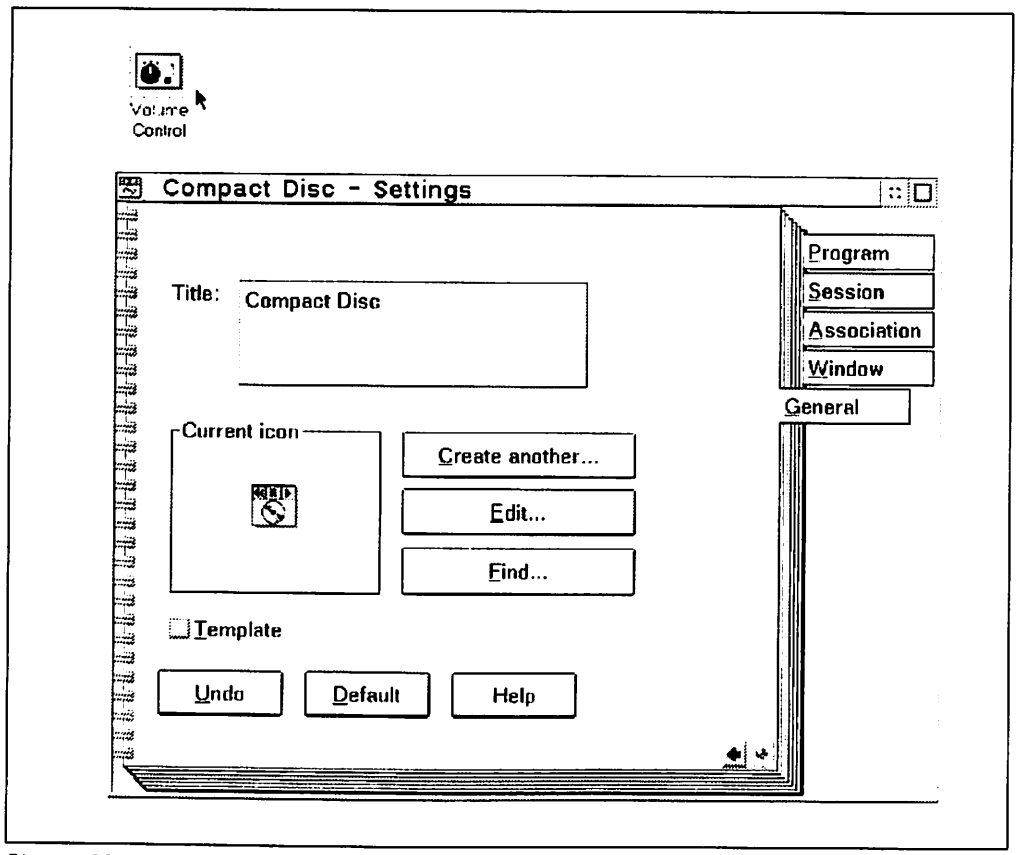


Figure 92. Workplace Shell - Dragging an Icon to the General Page

Open the **Settings** notebook of any object, and select the **General** page. Drag the icon of another object and drop it on the **Current Icon** field. This is shown in Figure 92 and Figure 93 on page 125.

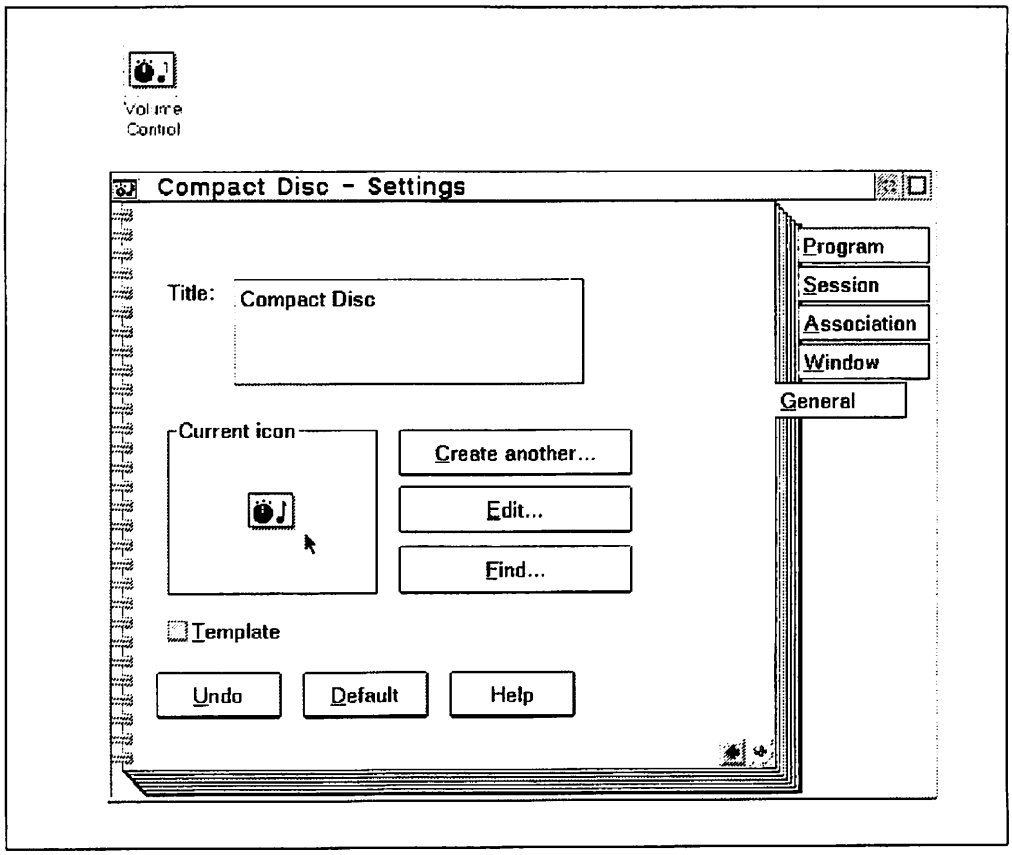


Figure 93. Workplace Shell - Changing an Icon by Drag/Drop

The icon is immediately changed, as shown in Figure 94.

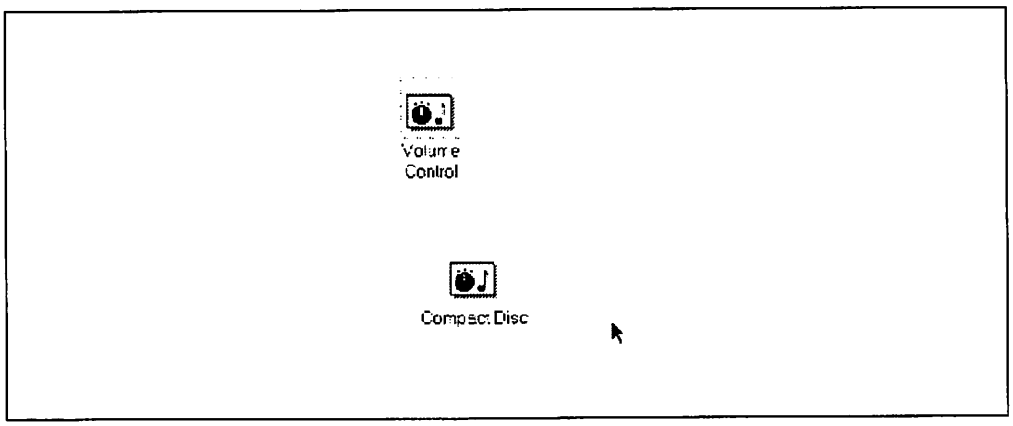


Figure 94. Workplace Shell - Changed Icon

To restore the original icon for the program object, click on **Default**.

9.3 Automatic Lock-up on System Startup

The security of an OS/2 2.1 workstation has been improved with the option to lock the system automatically on system startup.

Previously, it was possible to software-lock the system, but intruders could still avoid this software-lock feature by rebooting the system. The automatic lockup enhancement closes this security loophole.

To select this option, open the **Settings** notebook of the desktop, and display the **Lockup** page. Select the **Lock on startup** check box, as shown in Figure 95.

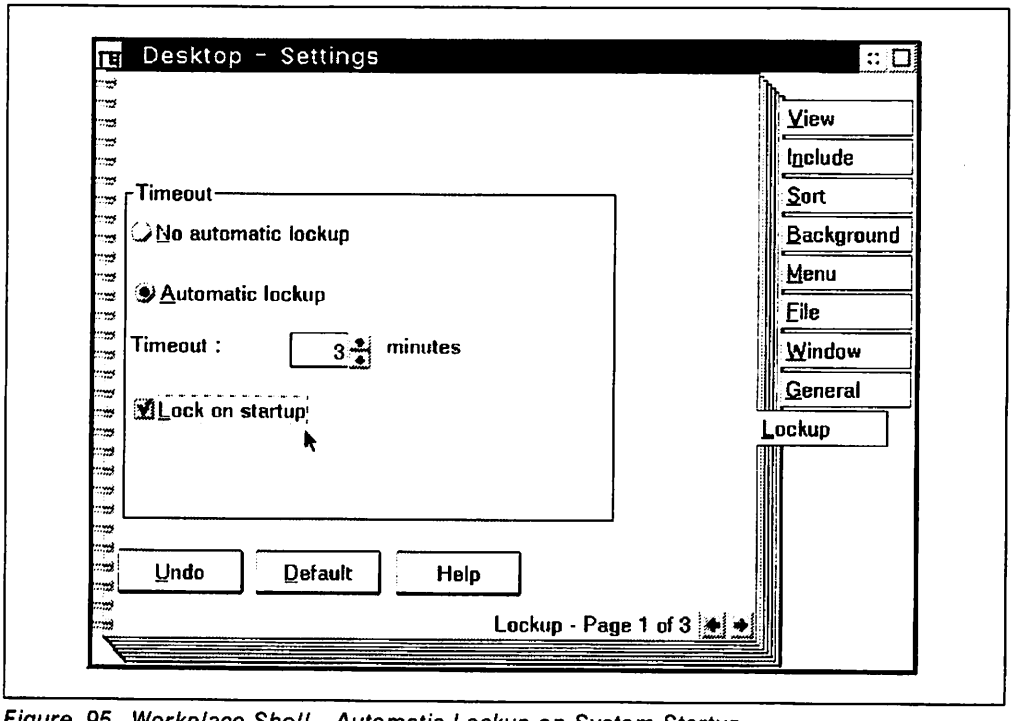


Figure 95. Workplace Shell - Automatic Lockup on System Startup

Warning

Use this function with care. If you forget your password, then there is no way to reset the password except by reinstalling OS/2 2.1, since the system will always prompt for a password even if rebooted.

9.4 Improved INI File Handling

Much of the configuration information of the Workplace Shell is held in the OS2.INI and OS2SYS.INI files in the \OS2 directory. (Workplace Shell information is also held in the directory structure and extended attributes of the \DESKTOP directory and of its subdirectories).

The handling of these INI files has been improved by rewriting the PMSHAPI modules as 32-bit code, and by changing the way INI files are handled. The internal structure of the INI files is not externally published, but has not changed (though it did change temporarily in the March 93 OS/2 2.1 beta).

The OS/2 Profile function APIs (Prf*) have not changed and can still be used to access the INI files.

INI files are now held in memory to provide faster access to the information stored in them whilst OS/2 2.1 is running. Any changes to the INI files are saved automatically by the system at frequent intervals, and also at system shutdown.

Chapter 10. Print Subsystem Enhancements

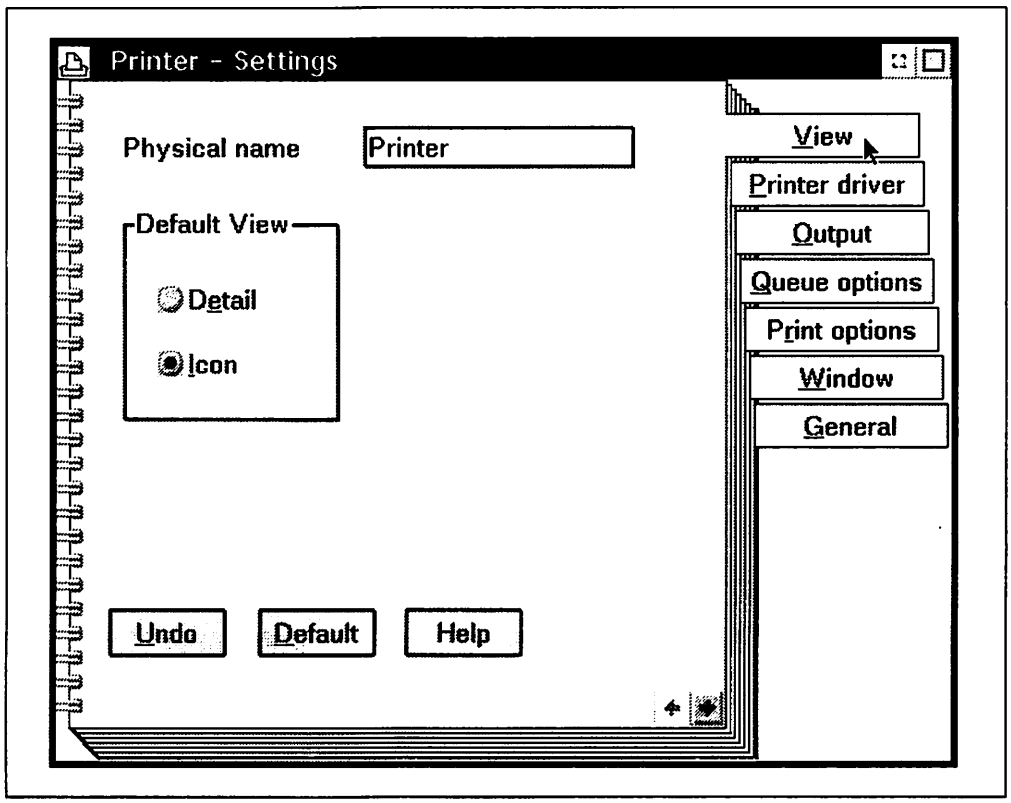


Figure 96. New Visual Representation of the Settings Notebook

In *OS/2 Version 2.0 - Volume 5: Print Subsystem*, we described the configuration and management of the OS/2 2.0 print subsystem. We also provided descriptions of the print subsystem components and their interaction.

This chapter will focus on the functional enhancements and improved configuration and management dialogs of the OS/2 2.1 print subsystem. In addition to the functional enhancements there have also been some visual changes to the intuitive interface. The most notable of these is shown in Figure 96 You will notice that the Settings "Notebook" now actually looks like a notebook. This will make it easier to understand what we are talking about when we say something like "..the Output tab in the Settings notebook of the printer object."

This chapter should be used in conjunction with *OS/2 Version 2.0 - Volume 5: Print Subsystem*.

10.1 Printer Drivers

Most of the changes in the print subsystem are in the area of printer drivers. The following have enhanced function and/or ease of use:

- Printer driver installation
- Listing printers and printer drivers
- Printer driver packaging
- Printer driver versions

10.1.1 Printer Driver Installation

In *OS/2 Version 2.0 - Volume 5: Print Subsystem*, we described how to install printer drivers using the context menu of an existing printer driver object. This method still exists, but a second, easier method is now available. Figure 97 shows the original method of installing a printer driver using the context menu of a printer driver object and the **Install...** option (see 1). Notice that below the **Default printer driver** container there is a new option button labeled **Install new printer driver...** (see 2). Clicking on this button is more intuitive and it will have the same result as using the install option in the context menu. In either case the **Install New Printer Driver** dialog shown in Figure 98 on page 131 will be displayed.

Note

The Install new printer driver... option will only appear in the Create a Printer dialog. It will not appear on the Printer driver page in the printer object settings. You can still install new printer drivers from the Printer driver page by using the context menu of an existing printer driver object. The enhanced Install New Printer Driver dialog will then be displayed.

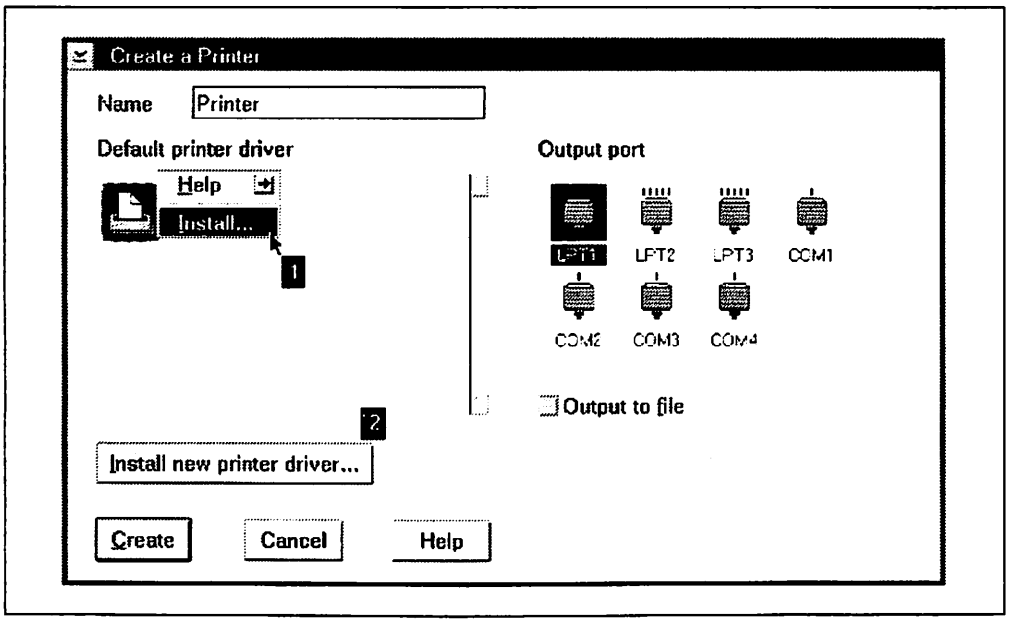


Figure 97. Ways to Install a New Printer Driver when Creating a Printer Object

The **Install New Printer Driver** dialog box has been enhanced (see Figure 98 on page 131). It lists all of the printers that the OS/2 printer drivers support. It also provides an option to list printer drivers from other sources.

To install a printer driver(s) that ships with OS/2:

1. Click on the desired printer driver(s) in the **Printer driver** container.
2. Click on **Install**.
3. Accept or change the **Directory** dialog (see Figure 99 on page 132 and Figure 100 on page 132).
4. Click on **OK** to install the printer driver.
5. Click on **OK** to confirm the installation or respond to any error.

Notes

If the driver for the printer you are installing is already installed you will get a message asking you if you want to replace it.

If you installed the driver using the Install... from the context menu of an existing printer driver then the Install New Printer Driver dialog will not close after installation. You must click on Cancel to do this.

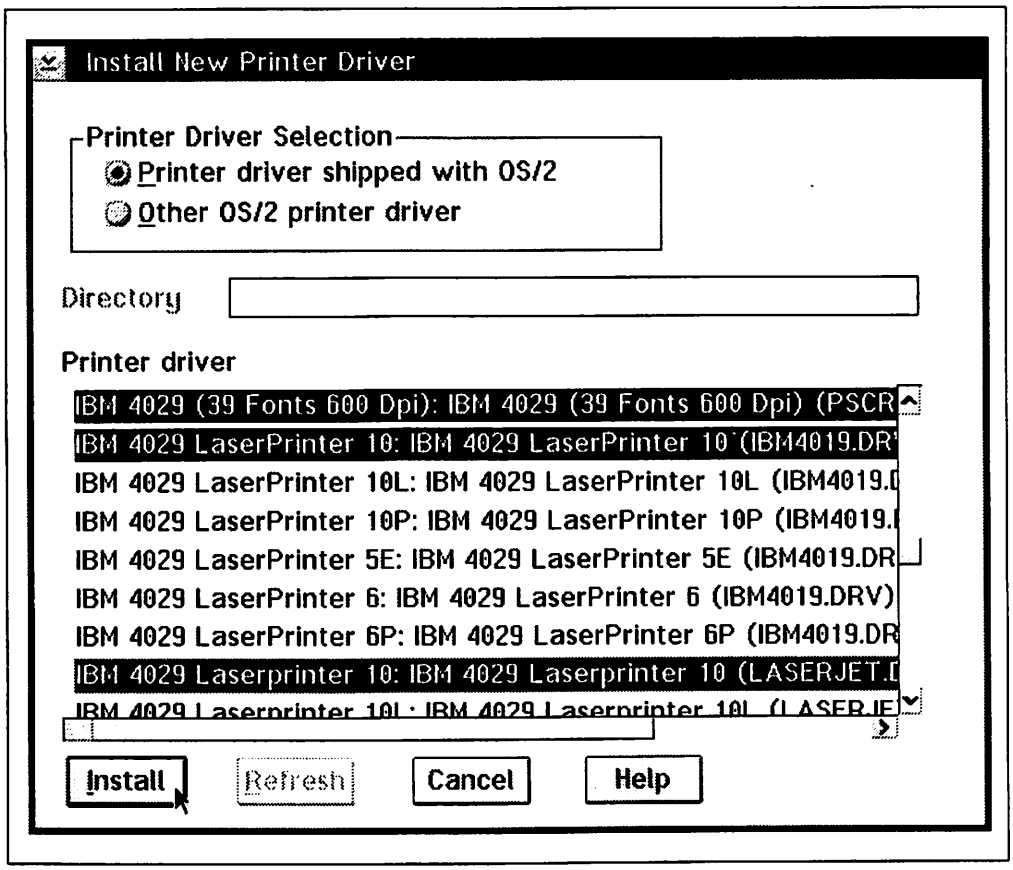


Figure 98. The Enhanced Install New Printer Driver Dialog

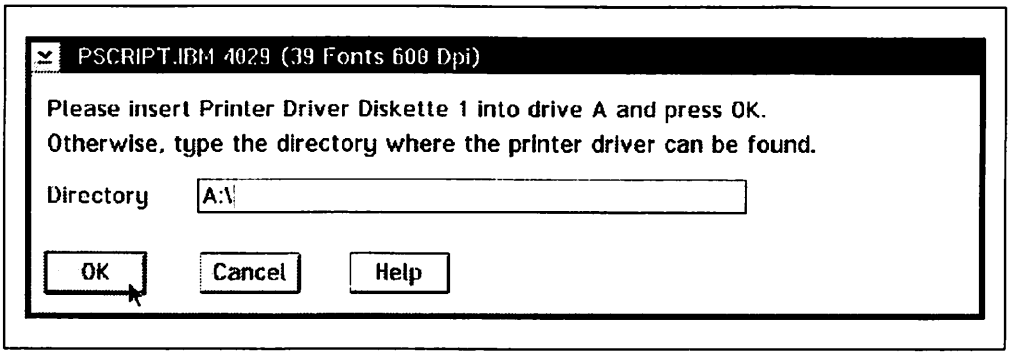


Figure 99. The Printer Driver Prompt if You Installed from Diskette

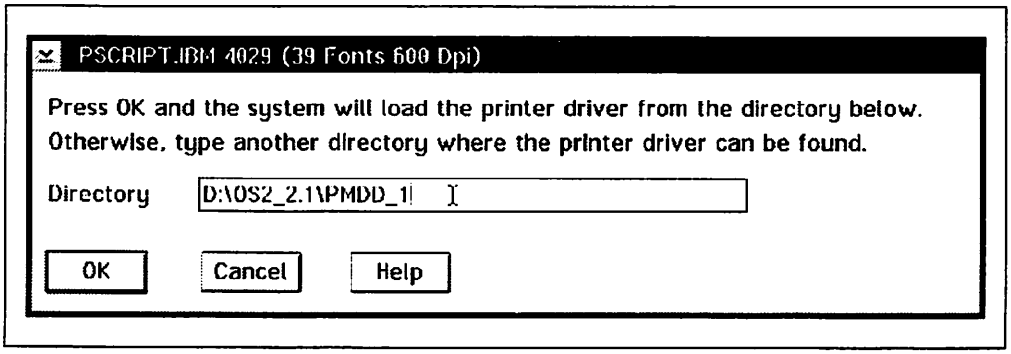


Figure 100. The Printer Driver Prompt if You Installed from Other Than Diskette

To install an OS/2 printer driver(s) from another source:

1. Click on **Other OS/2 printer driver**.
2. Type the path of the printer driver in the **Directory** field.
3. Click on **Refresh**.
4. Click on the desired printer driver(s) in the **Printer driver** container.
5. Click on **Install**.
6. Click on **OK** to confirm the installation or respond to any error.

See Notes on page 131 with regard to some additional dialogs that might be displayed.

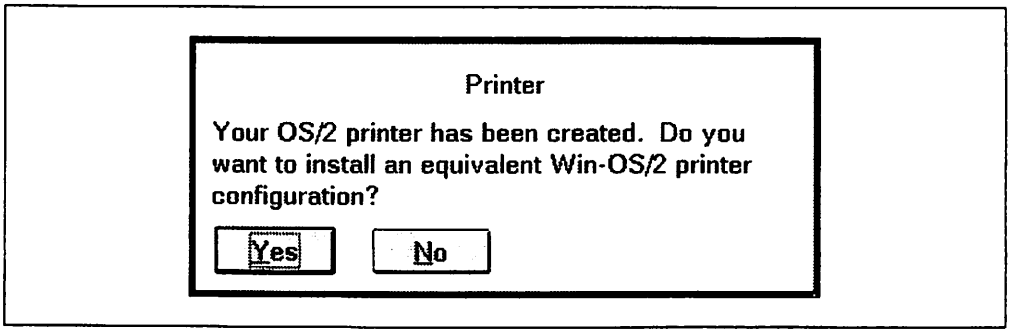


Figure 101. The Prompt to Install the WIN-OS/2 Printer Driver

If you are *creating* a new printer object then you *may* get a prompt asking you whether you want to install the equivalent WIN-OS/2 printer (see Figure 101). If you click on **Yes** then you will get a similar dialog box to those that are displayed in Figure 99 and Figure 100 in order to install the WIN-OS/2 driver.

Changes in Recommendations

In *OS/2 Version 2.0 - Volume 5: Print Subsystem* there are some notes that recommend that you *not* install WIN-OS/2 printers at the same time as Presentation Manager printers. With the new dialog box allowing for the WIN-OS/2 printer drivers to be in any path, we now recommend that you *do* install the WIN-OS/2 printers if you are prompted by the PM installation.

Figure 99 on page 132 and Figure 100 on page 132 display an install path based on the source of your OS/2 installation. This information is in the OS2.INI file. The following REXX procedure will allow you to change the path. If you change the path from A: then the directory PMDD_x, where "x" is the diskette number, will be inserted at the end of the path by the system. You must copy the printer driver diskettes to (sub)directories of these names. If you execute the program without a parameter it will query and show you the current install path. If you enter a parameter such as D:\PDRIVERS it will change the printer driver Directory prompt to D:\PDRIVERS\PMDD_x where "x" is the diskette number.

```
/* Sample Procedure to query or change the printer driver directory */
Parse Upper Arg Path .
Inifile = 'USER'
App = 'PM_INSTALL'
Key = 'PDR_DIR'
Key2 = 'MEDIA'
call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'
if Path = ""            /* if no path specified, query path and display */
then do
     Path = SysIni(Inifile, App, Key)
     if path = 'ERROR:'
     then do
          say "The key '"key"' and/or the application '"App"'"
          say "was not found in the '"Inifile"' inifile. Check the"
          say "inifile parameter in this CMD file and try again."
          end
     else say "The actual printer driver path is:" Path
     end
else do
     result = SysIni(Inifile, App, Key)
     if result = 'ERROR:'
     then do
          say "The key '"key"' and/or the application '"App"'"
          say "was not found in the '"Inifile"' inifile. Check the"
          say "inifile parameter in this CMD file and try again."
          end
     else do
          if left(Path, 2) = 'A:'
          then Media = 'DISKETTE'
          else Media = 'REMOVABLE'
          if right(Path, 1) = '\'
          then Path = left(Path, Length(Path) - 1)
          result = SysIni(Inifile, App, Key, Path)
          result = SysIni(Inifile, App, Key2, Media)
          say "The printer driver directory was changed to" Path
          end
     end
```

Figure 102. Sample Procedure to Query or Change the Printer Driver Directory

10.1.2 Listing Printer Drivers

In *OS/2 Version 2.0 - Volume 5: Print Subsystem* there is a section that shows you how to list the printers and their associated printer drivers that ship with OS/2 2.x. The location of these lists is on the first diskette of the OS/2 2.0 Printer Driver Diskettes. These files are now also placed on the install drive during OS/2 2.1 installation. They are located in the \OS2\INSTALL directory and are named PRDESCR.LST and PRDRV.LST. With the changes in the printer driver install dialog, the need for these files is not as critical, but, if you want to print a list of all the available printers and printer drivers then you could open these files in the Enhanced Editor and use the print option in the file menu. In the example below we have listed these files as they appear in the initial release of OS/2 2.1. The location and printers supported by these drivers has changed since OS/2 2.0 and will most likely continue to change in the future. Therefore you should always display the files on your system for an accurate list.

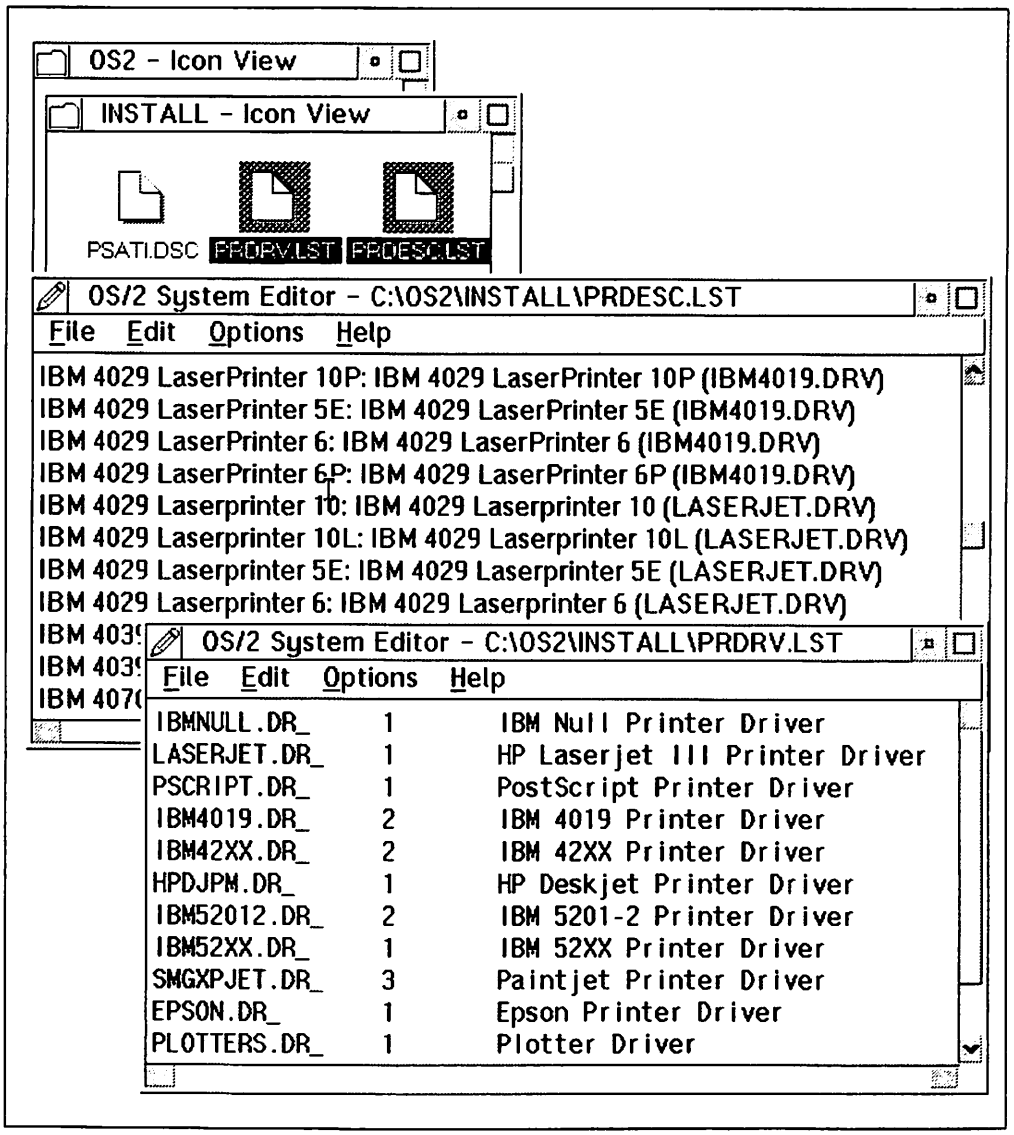


Figure 103. Listing Printers and Printer Drivers Supported by OS/2 2.1

10.1.3 Printer Driver Version Changes

The following table shows the printer drivers and their various versions. The older versions of these printer drivers can be replaced with these newer versions. See the section on reinstalling a printer driver in *OS/2 Version 2.0 - Volume 5: Print Subsystem* for instructions on how to do this.

Table 23. Current OS/2 Printer Driver Versions

Printer Driver	Initial Release	Service Pak	OS/2 2.1
Epson (EPSON.DRV)	13.342	13.354	13.384
IBM 4019 (IBM4019.DRV)	1.281	1.281	1.354
IBM 42xx (IBM42XX.DRV)	1.281	1.309	1.354
IBM 52xx (IBM52XX.DRV)	1.281	1.284	1.289
IBM 52012 (IBM52012.DRV)	1208	1208	1208
HP Laserjet (LASERJET.DRV)	1.3.342	1.3.353	1.3.385
HP PaintJet (SMGXPJET.DRV)	3.9.p2	3.9.p2	3.9.p2
PostScript (PSCRIPT.DRV)	13.342	13.342	13.388
Plotters (PLOTTER.DRV)	13.342	13.353	13.380
HP DeskJet (HPDJPM.DRV)			13.380
IBM Null (IBMNULL.DRV)	1208	1208	1208

These new OS/2 2.1 printer drivers are in a packed format. OS/2 2.1 can install them as is but they must be unpacked if you want to use them with an earlier version of OS/2. To unpack a driver, create a directory on your hardfile and then execute the UNPACK command from an OS/2 Command Prompt. For example, if you are using diskettes and create a HP directory on C:

```
UNPACK A:LASERJET.DR_ C:\HP
```

You can then install it directly from the target directory or copy it to a diskette for installation purposes.

Warnings!

The new PostScript printer driver is 32-bit and unique to OS/2 2.1! In the past we have said that you can use the OS/2 2.x printer drivers for previous release of OS/2. Do not attempt to use this driver with *any* previous release!

Do not UNPACK an entire diskette into the same directory. Unpack one driver at a time. Packed diskettes may contain file names that, when unpacked, are duplicates and will overwrite each other. Do not unpack the WIN-OS/2 drivers. They can be used as is.

10.2 Spooler Settings

In addition to the Spool path tab, two new tabs have been added to the Settings notebook of the Spooler object. They are the Print priority and General tabs.

10.2.1 Print Priority Setting

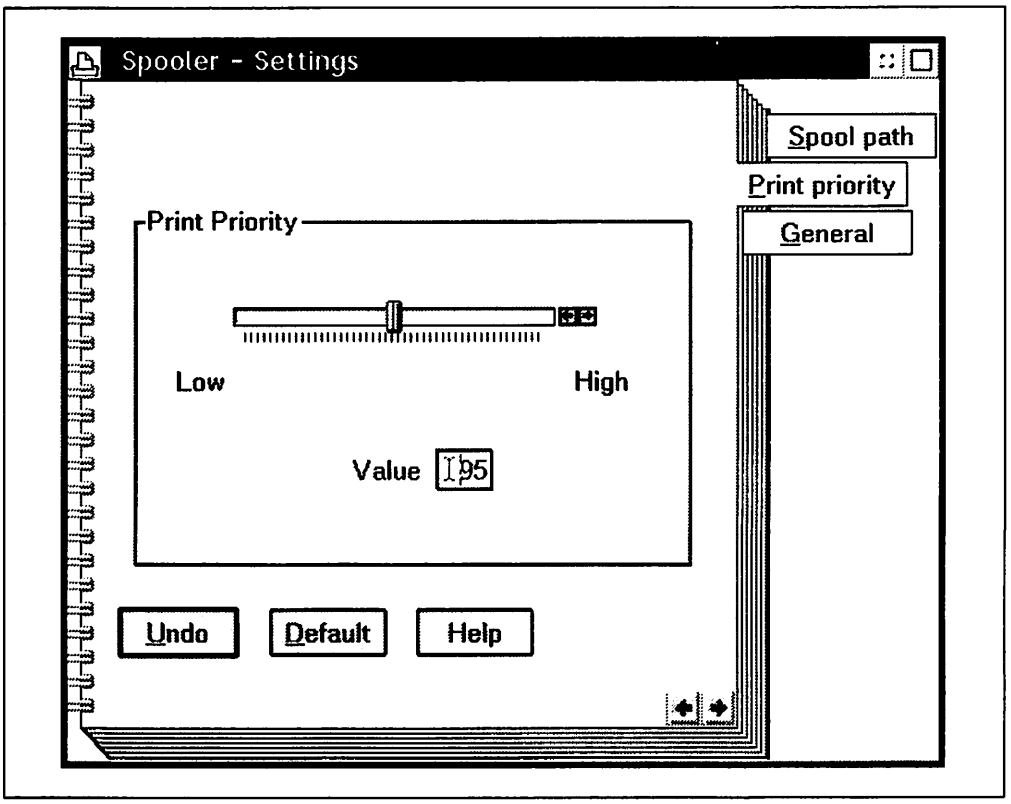


Figure 104. Setting the Spooler Print Priority Level

The Print priority setting allows you to balance and optimize the system's performance between printing and running applications. The default setting is 95 and the range is from 1 to 189. To better understand how this setting affects performance we set up a small test. We used a PS/2 Model P75 logged onto a LAN Server network. CorelDraw! was used to print and then open a new file while printing. The file we printed was DART.CDR and the file we opened was ALLFONTS.CDR. The other programs that were running in the last scenario were the System Clock in second-hand analog mode and Pulse. As expected, the print priority setting directly effects the system performance. The following table shows our results.

Table 24. Spool Priority Effect on a System (Print Time/Open Time) (min:sec)

Spooler Priority	Print Only	Print and Open File	Print, Open and Run Programs
189	0:50/0:00	0:55/0:35	0:55/0:35
95	0:50/0:00	1:08/0:27	1:13/0:27
1	0:52/0:00	1:15/0:23	2:45/0:23

In the following figures we monitored the system's progress for our last test scenario. The graphs chart (A) the creating of the print job, (B) the point at which the job is in the queue and we start to open another file, (C) the point at which the file is opened and displayed, and (D) the point at which the print job clears the queue. The times for these charts can be found in the last column of the previous table. A to D is the print time and B to C is the time to open the file.

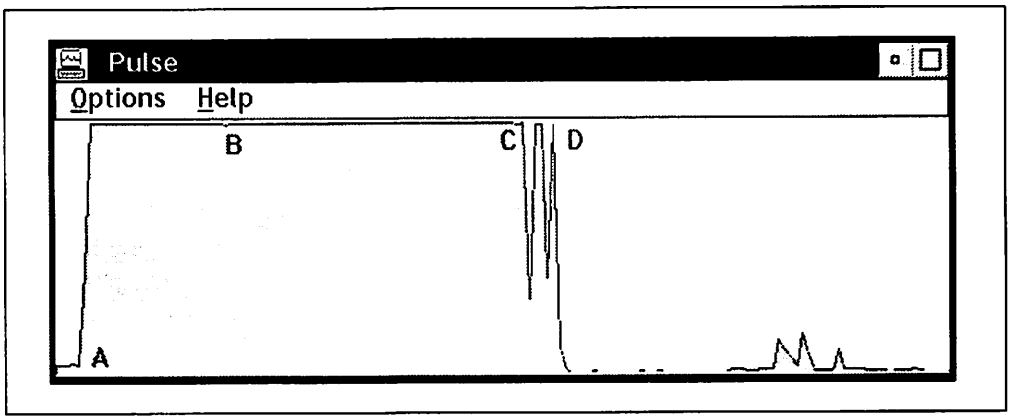


Figure 105. System Activity with a Spool Priority of 189

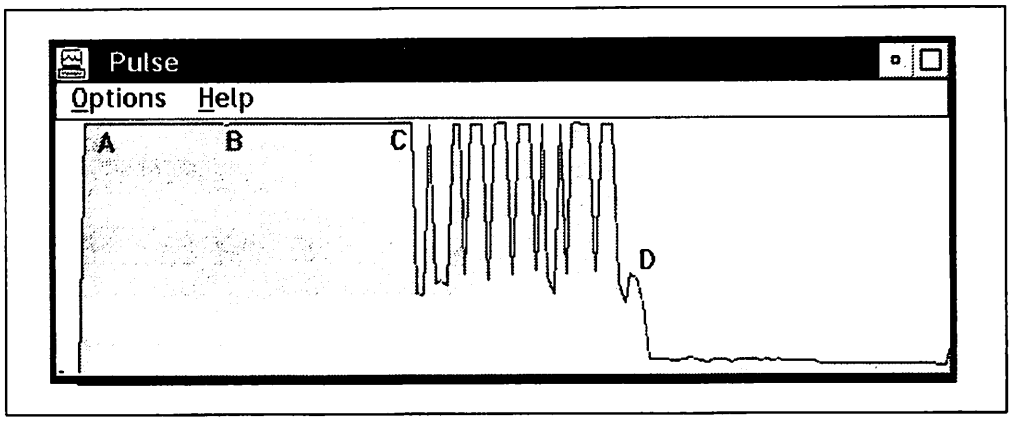


Figure 106. System Activity with a Spool Priority of 95

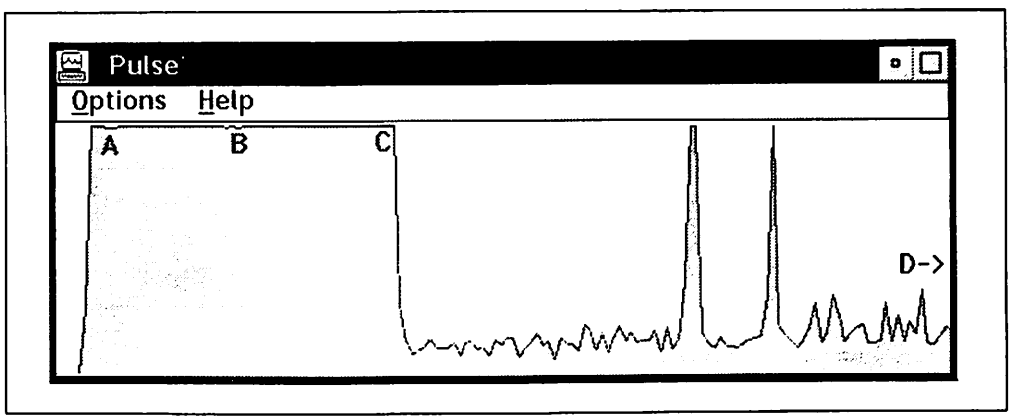


Figure 107. System Activity with a Spool Priority of 1

10.2.2 Spooler General Setting

The Spooler General setting is new and is like the General setting in other OS/2 objects. It allows you to change the name of the Spooler object and its associated icon. We have included the Spooler path in this section to show you the difference between the Spool folder (directory) name and the Spooler object (icon) name.

Unlike printer objects, where you can't change the folder (queue) name once the system generates it, the Spool folder name can be changed on the Spool path page of the Spooler object settings. At install time the system sets the folder name to Spool and the object name to Spooler. The following figures show our desktop and folder structure after we changed both names in the settings.

Remember

Even though you can change the Spool folder name in the settings of the Spooler object, you cannot access the contents of the Spool folder through the Spooler object. The contents of the Spool folder should only be accessed through the printer object(s) as they contain the queued print jobs.

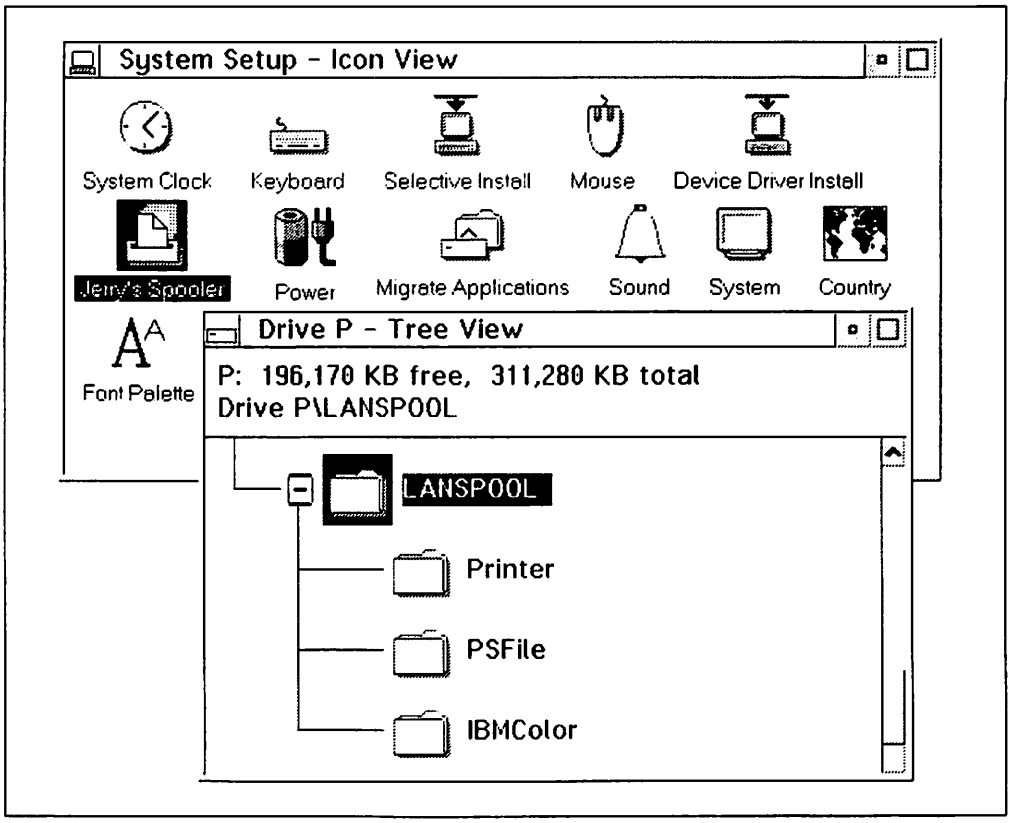


Figure 108. Spool Folder Name versus Spooler Object Name

Warning!

If you change the Spool path (directory) name make sure the name conforms to DOS FAT file conventions (eight characters). The system will allow a complex name to be entered but you will not be able to print from DOS or WIN-OS/2 as they will not see the spool directory.

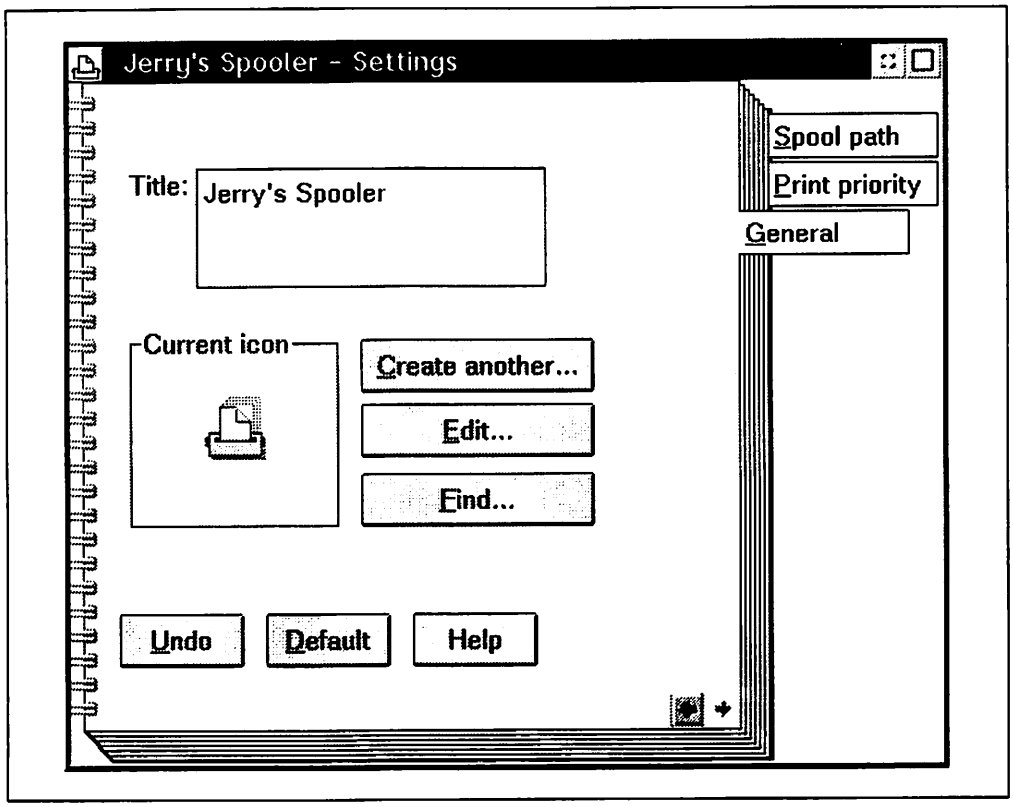


Figure 109. Changing the Name of the Spooler Object

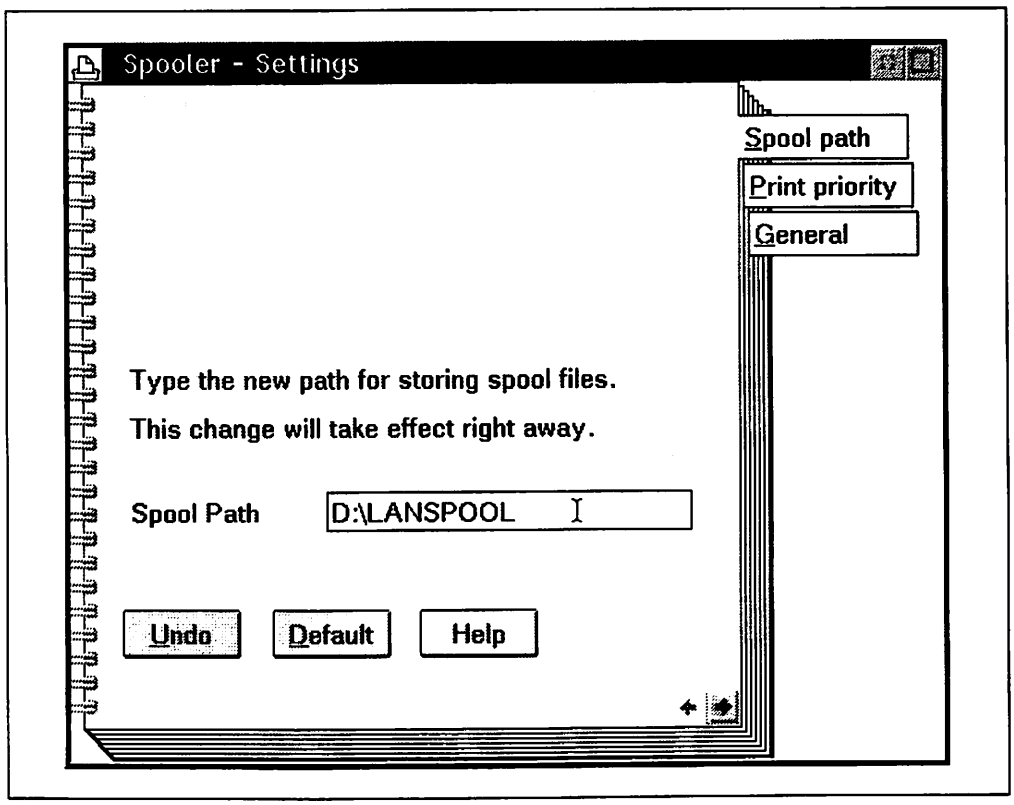


Figure 110. Changing the Location and Name of the Spool Folder

10.3 Shared Access of Parallel Port

Checking the Share access check box allows multiple DOS applications to share a parallel port. This may be required for those DOS applications that try to access a port during initialization. The most frequent of occurrence is when you are required to use a security device attached to the parallel port.

If the port is already in use, error SYS1799 will appear on the screen. Checking the Share access check box will prevent the error from occurring. If Share access is enabled and you get intermixed output from DOS applications, disable Share access. The default is to have Share access unchecked (disabled).

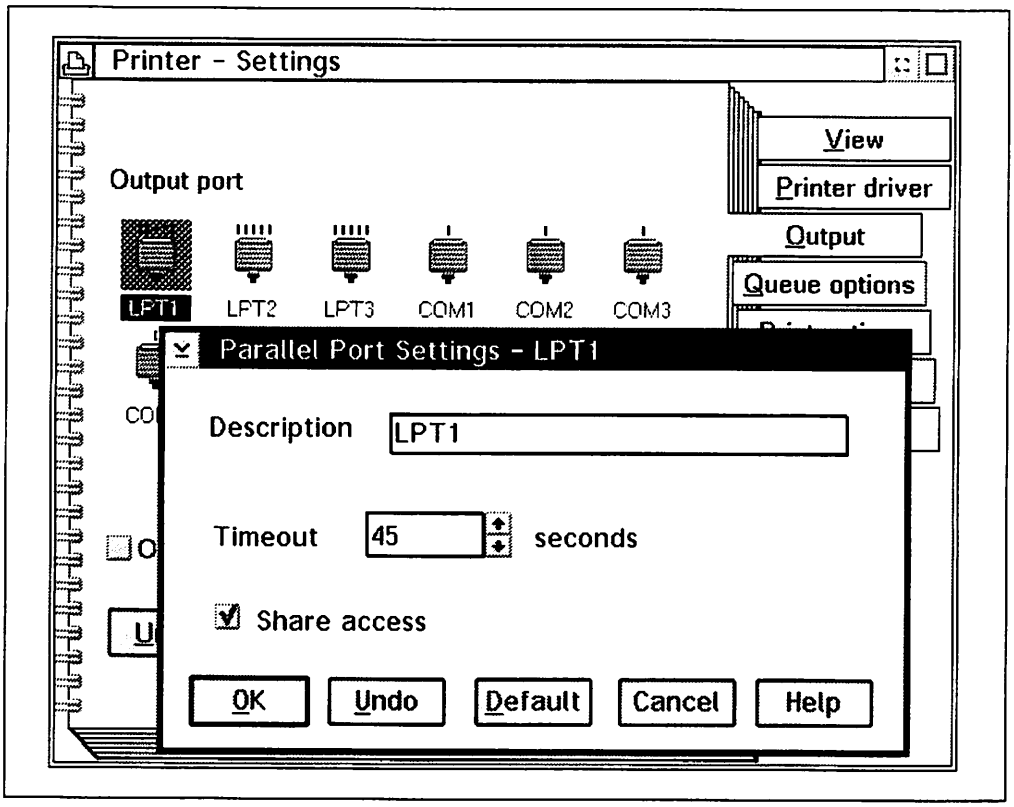


Figure 111. Setting Shared Access of a Parallel Port

10.4 Redirection of Ports

In OS/2 2.0 you were only able to redirect from an *assigned* LPT port to another assigned port. OS/2 2.1 now allows you to redirect from *any* LPT port to any assigned port. Aside from the normal use of port redirection this new implementation allows us to create a simple scenario in which all WIN-OS/2 spooling is handled by OS/2. Remember that by default all WIN-OS/2 serial (COMx) print jobs are spooled in WIN-OS/2 and do not go through the OS/2 spooler. This creates a problem if you are trying to monitor both LPT and COM printers on the same machine. To spool serial WIN-OS/2 print jobs through OS/2 do the following:

- On the OS/2 desktop:
  1. **Create** an OS/2 printer object for the serial printer and assign it to **COM1**.
  2. Answer **Yes** if prompted to install equivalent WIN-OS/2 printer.
  3. Open the printer object settings of the OS/2 printer object.
  4. Click on the **Output** tab.
  5. Display the context menu of **LPT2** (mouse button 2).
  6. Click on the arrow after **Redirection**.
  7. Click on **COM1**.
- In WIN-OS/2:
  1. Access the printer settings through the **Control Panel**.
  2. Use **Connections** to assign the equivalent printer to **LPT2.OS2**.

Note that we used COM1 and LPT2 in our scenario. You should use the actual COM port that your printer is attached to and then any LPT port that is not being used by the system. This scenario will also work for pooled COM printers.

10.5 OS/2 Printer Object and Equivalent WIN-OS/2 Printer

OS/2 determines if there is an equivalent WIN-OS/2 printer driver for an OS/2 printer driver by using the DRVMAP.INF file in the \OS2\MDOS\WINOS2\SYSTEM folder. If the default printer driver in the OS/2 printer object you are creating is listed there, you will get a dialog box asking you if you want to install the equivalent WIN-OS/2 printer (see Figure 101 on page 132).

Sometimes when you create an OS/2 printer object you will *not* be prompted to install an equivalent WIN-OS/2 printer driver. This is because the default printer driver for the OS/2 printer object you are creating is not listed in the DRVMAP.INF file. If you know that there is a printer driver in WIN-OS/2 that you can use with your printer then you can use the Control Panel of WIN-OS/2 to install it separately. This method is acceptable if you only have to install a few of these printers on a few PCs.

If you have many PCs with a printer(s) that is not listed in the DRVMAP.INF file then you might want to consider editing the file to include it. The file could then be available over a network and copied over the default file that comes with OS/2. In addition to saving time, the main reason to do this edit is to ensure that printers are installed correctly in both the OS/2 and WIN-OS/2 environment. By using the DRVMAP.INF file the system can automatically do the WIN-OS/2 install for you.

10.5.1 Using the Default DRVMAP.INF File

Let us use the scenario of the IBM 4029 Model 10. You may choose to use it in the PCL emulation. If you create an OS/2 printer object with the IBM 4029-10 (LASERJET) as the default printer driver, however, you will not be prompted to install a WIN-OS/2 equivalent. The result will be what you see in Figure 112. You will have a printer object on your desktop with no corresponding printer in WIN-OS/2.

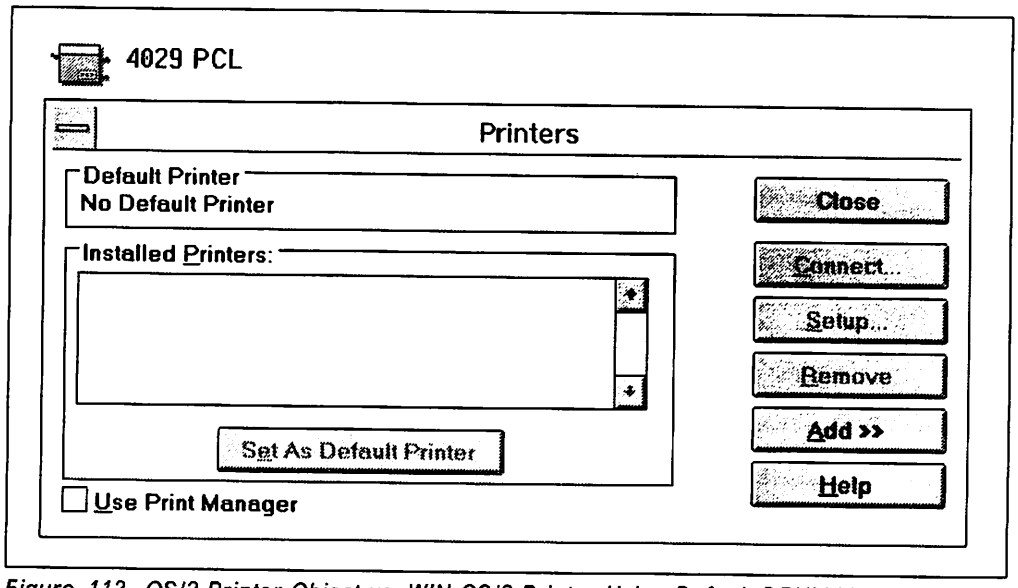


Figure 112. OS/2 Printer Object vs. WIN-OS/2 Printer Using Default DRVMAP.INF

10.5.2 Editing the DRVMAP.INF File

If you edit the DRVMAP.INF file to include an entry for the printer, then you will be prompted and a corresponding printer will be installed in WIN-OS/2. The entry we added to the DRVMAP.INF file was:

LASERJET.IBM 4029 Laserprinter 10=HP LaserJet Series II (PCL / HP LaserJet)

This entry is divided into three parts and in our example they are:

- The name of the OS/2 printer driver file (LASERJET)
- The name of the OS/2 printer in the driver file (.IBM 4029 Laserprinter 10)
- The name of the WIN-OS/2 printer (=HP LaserJet Series II (PCL / HP LaserJet))

To find out what names to use for the printer we used the PRDESC.LST file along with the DRVMAP.INF file. You will also need to know what emulation your printer uses. In our case we knew that the printer emulated a LaserJet but we had to read the documentation supplied by the manufacturer to determine *which* LaserJet emulation to use. You can now edit the DRVMAP.INF file as follows:

1. Make a backup copy of the DRVMAP.INF file.
2. Open the PRDESC.LST and DRVMAP.INF files in an editor.
3. Scroll through the PRDESC.LST file to find the entry for the OS/2 printer you want to install.

4. Scroll through the DRVMAP.INF file to find an entry for the emulation mode of the printer you want to install.

   Note that the file name of the printer driver in both entries *must match*. In this case the file name refers to the name without the extension (LASERJET). (See Figure 113.)

5. Duplicate the DRVMAP.INF entry by using copy and paste. (See Figure 114.)
6. Copy the OS/2 printer name in the PRDESC.LST file entry. (See Figure 114.)
7. Paste the OS/2 printer name from the PRDESC.LST file over the OS/2 printer name in the DRVMAP.INF file entry. (See Figure 115 on page 144.)
8. Save the DRVMAP.INF file.

Hint: Look in the \OS2\MDOS\WINOS2\SYSTEM\CONTROL.INF file for the WIN-OS/2 printer name if the DRVMAP.INF does not list it.

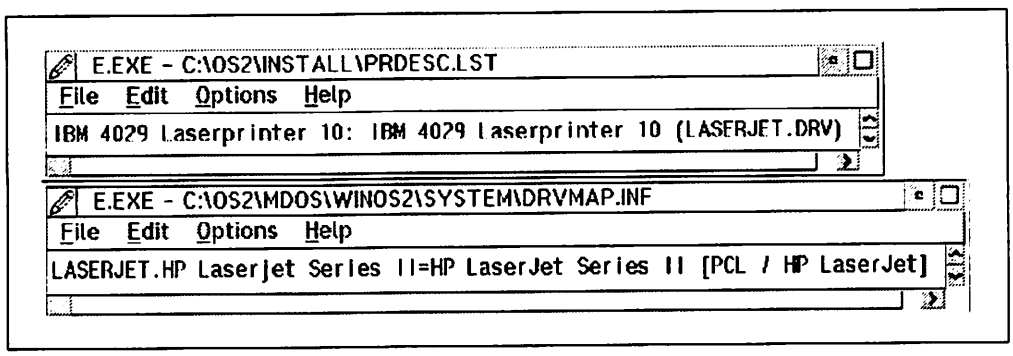


Figure 113. The DRVMAP.INF File and the PRDESC.LST File

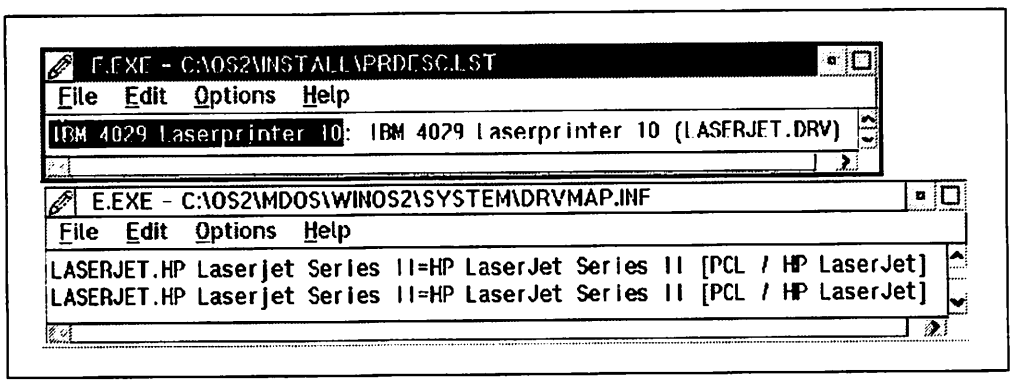


Figure 114. Copying the OS/2 Printer Name from the PRDESC.LST File

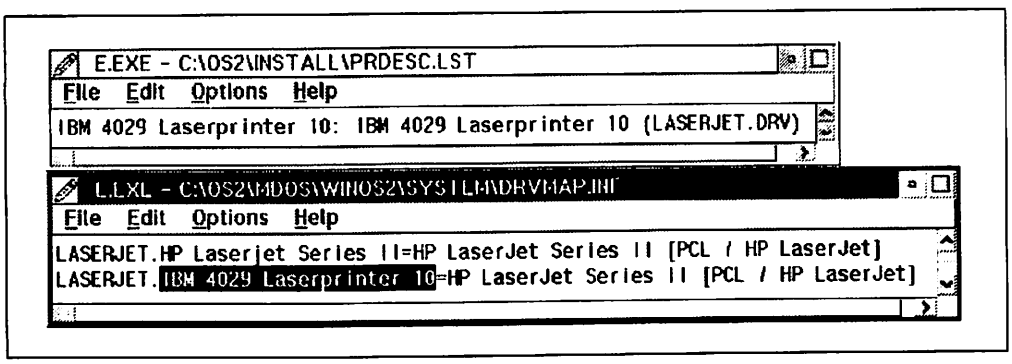


Figure 115. Pasting the OS/2 Printer Name in the DRVMAP.INF File

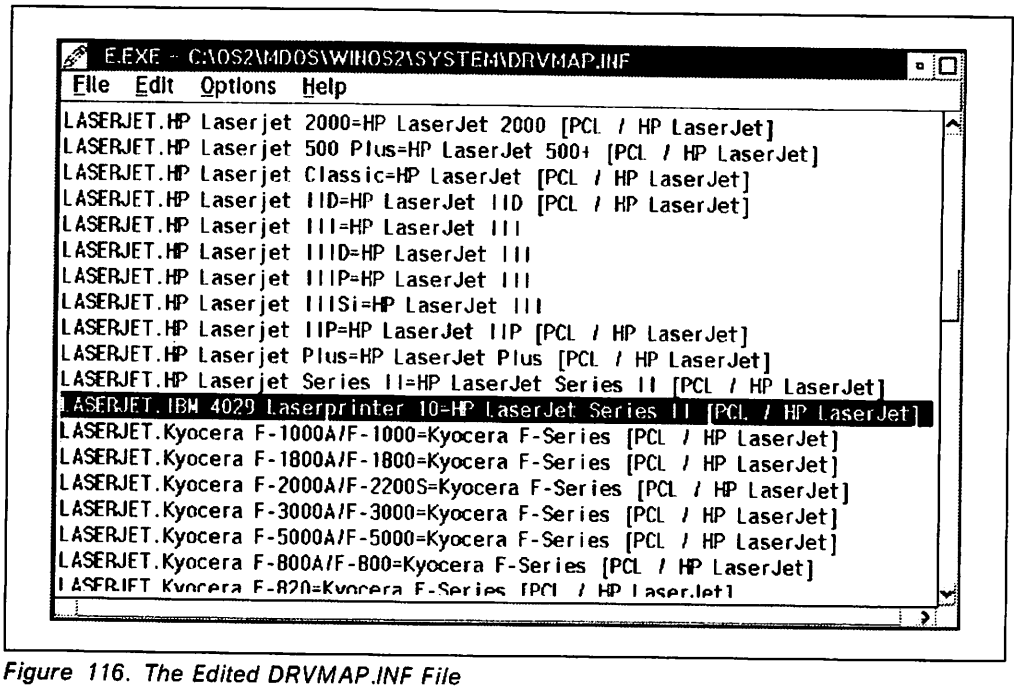


Figure 116. The Edited DRVMAP.INF File

10.5.3 Using the Edited DRVMAP.INF

After we edited the DRVMAP.INF to include the IBM 4029 in LaserJet emulation, we deleted the IBM 4029 PCL printer object. We then created an OS/2 printer object using the IBM 4029 Laserprinter 10 (LASERJET.DRV). The prompt to install the WIN-OS/2 printer appeared this time and we said "Yes." Figure 117 on page 145 shows the resulting installation for the printer in both OS/2 and WIN-OS/2.

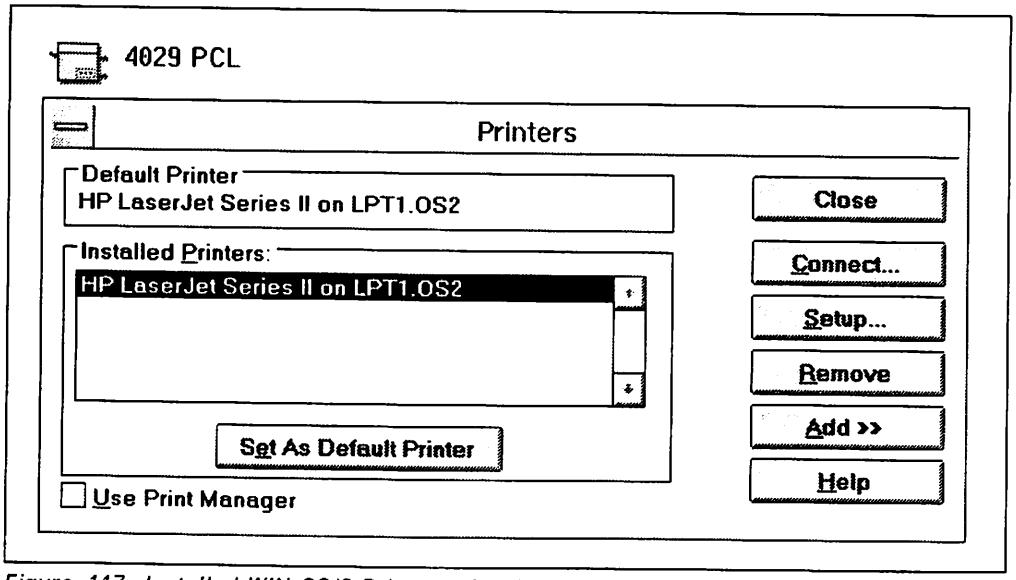


Figure 117. Installed WIN-OS/2 Printers after Using the Edited DRVMAP.INF File

10.6 Display Modes vs. Suspension of Print Jobs

In *OS/2 Version 2.0 - Volume 5: Print Subsystem* we stated the following:

> **Note**
>
> It should also be noted that in some resolutions, when printing from a full screen WIN-OS/2 session, printing is suspended if you switch back to the Workplace Shell while the print job is still spooling.

We have been able to do some additional testing to expand on this note. In OS/2 2.0 we found that when you sent print jobs from WIN-OS/2 full screen sessions the spooler behaved as follows:

- If OS/2 2.0 was installed with VGA resolution then the spooling continued to work in the background when you switched back to the Workplace Shell.
- If OS/2 2.0 was installed with XGA resolution then spooling would suspend when you switched back to the Workplace Shell and only resume when you switched back to the WIN-OS/2 session.

This restriction has been eliminated in OS/2 2.1. When running either seamless or full screen WIN-OS/2 sessions in XGA mode, if you print from these sessions, you will be able to continue working with other programs from the Workplace Shell while jobs are spooling and printing.

Chapter 11. Enhancements for Laptops and Notebooks

OS/2 2.1 contains new functions for the user of portable computer systems, such as laptops and notebooks, namely:

- Support for the Advanced Power Management (APM) specification
- Personal Computer Memory Card International Association (PCMCIA) enabling for I/O.
- VGA large cursor on LCD screens
- Trackpoint II support

These functions are described in this chapter.

11.1 Advanced Power Management (APM) Support

This section describes the APM (Advanced Power Management) specification and the APM support provided by OS/2 2.1.

11.1.1 APM Specification

Advanced Power Management (APM) was introduced by Intel Corp. and Microsoft Corp. as a platform-independent software specification to extend the battery life of portable computers. Thirty-six companies endorsed the APM specification. Under APM, the operating system communicates precise information to the firmware -- software that communicates between the operating system and hardware -- about the system's power usage. This information enables the firmware to make better and faster choices in the real-time task of saving power. The result is superior battery conservation, which enables users to work with their portable systems for longer periods of time.

The APM specification complements existing power management designs provided by hardware vendors. APM eliminates the need for PC manufacturers to write complex operating system-specific software drivers for power management that could slow the system or create incompatibilities. APM compliance extends the power-consciousness of the computer system beyond the firmware to include the operating system, drivers and applications. An APM driver mediates between the APM-compliant system BIOS and the operating system. It seeks information from the operating system and applications about their status and their needs to access hardware. Applications that are themselves APM-compliant can contribute even more power data to the APM driver, which passes this information to the firmware. The firmware, in turn, continually manages the hardware through a system vendor's unique built-in power-saving scheme. The additional data from the APM interface allows the firmware to make better and faster decisions about power savings.

11.1.2 Personal Computers with APM Support

Currently, the following IBM personal computers support the APM standard:

- IBM ThinkPad 300
- IBM ThinkPad 700 and 700C
- IBM ThinkPad 720 and 720C
- IBM PS/note Model N45 SL

A number of PCM portable computers also comply to the APM specifications.

11.1.3 OS/2 2.1 APM Support

OS/2 2.1 provides features to manage and track power consumption in battery-powered computers that support the Advanced Power Management (APM) specification.

If a computer supports the APM standard, the power object is automatically installed in the **System Setup** folder during the OS/2 2.1 installation process. If the Power object is not installed during the installation process, it can later be installed using Selective Install. Refer to 3.2, "Enhancements to the Selective Install Program" on page 18 for instructions on using the Selective Install program.

11.1.4 The Power Application

The power application provides you with the following features:

- A choice that sets power management to on or off
- A status window that shows whether your system is running from battery power or AC power
- A status windows that shows you the power level of the battery and the state of the battery charge
- A choice to automatically update the status window at user-selectable intervals
- A choice that conserves power by using partial power levels (Suspend function)

11.1.4.2, "Power - Settings" on page 151 explains how to set these features.

11.1.4.1 Power Status

To display the power status, start the power application from the **System Setup** folder. If your computer is currently running on battery power, a window similar to Figure 118 is displayed.

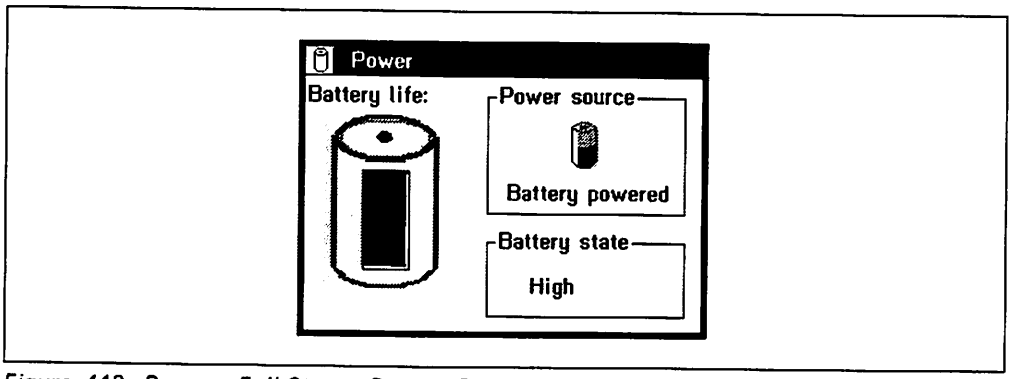


Figure 118. Power - Full Status, Battery Powered

If your computer is currently running on AC power, the status might look like Figure 119 on page 149.

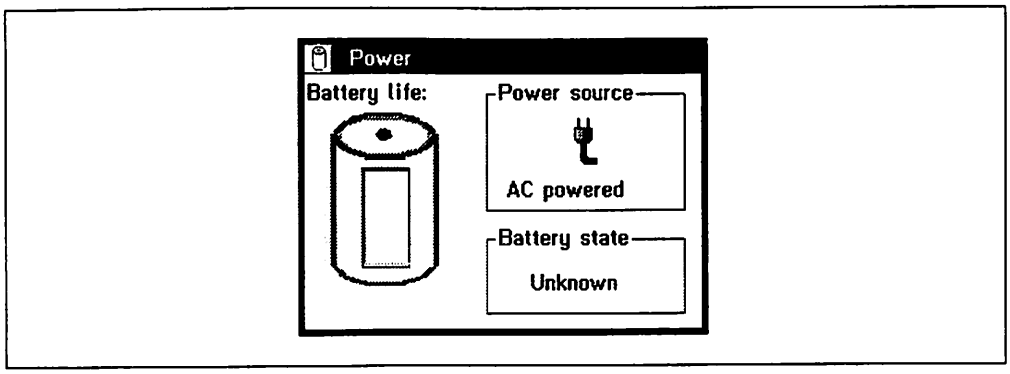


Figure 119. Power - Full Status, AC Powered

The following information is provided in the Power Status window:

1. Power source

   The power source information is displayed as either **Battery powered** (operating with a battery pack) or **AC powered** (operating with electric current). If the system cannot determine the power source, no Power Source information is displayed.

2. Battery life

   This information is displayed as a battery and power gauge graphic. The power gauge shows the power level of the battery compared to the capacity of the battery. When the shaded area of the gauge is at the top of the scale (100%), the battery is at full power. When it is at the bottom of the scale (0%), the battery is out of power. When the shaded area is dimmed, there is either no battery in the computer or the computer cannot provide battery information.

3. Battery state

   Either one of the following battery states is displayed:

   - **High** - The battery charge is high and you can continue using your system.
   - **Low** - The battery charge is reduced and you need to prepare to recharge the battery or switch to another power source, such as another battery or AC power.
   - **Critical** - The battery charge is depleted and you need to recharge the battery or switch to another power source.
   - **Charging** - The system is restoring the battery charge.
   - **Unknown** - The system cannot determine the battery state.

   Warning

   A **Critical** battery state could cause system failures or data loss. Recharge your battery or switch to another power source immediately!

The context menu of the power application shown in Figure 120 on page 150 gives you access to a number of functions explained below.

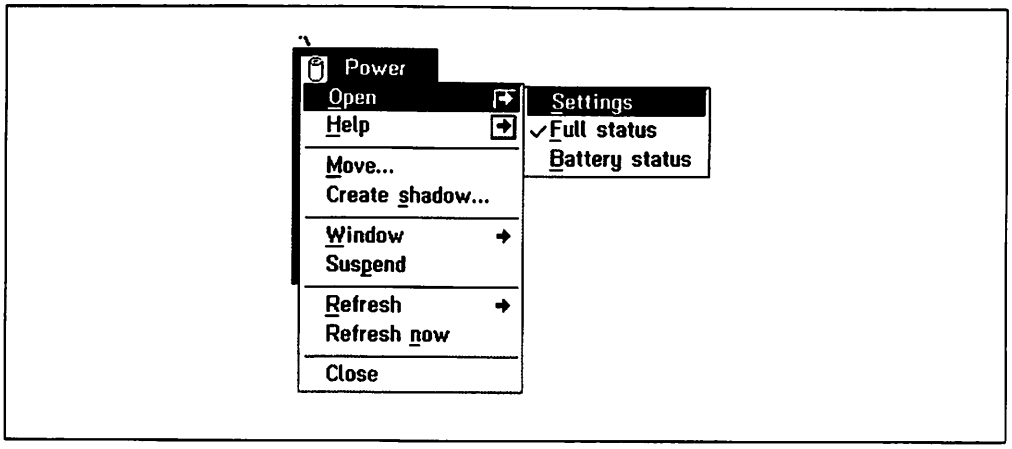


Figure 120. Power - Context Menu

- Full status or Battery status

  Alternatively to the **Full Status** shown in Figure 118 on page 148 and Figure 119 on page 149, a **Battery Status** can be selected. This view contains only the battery life information. Due to the reduced size of this view, it is handy to have the battery status always displayed. The Battery Status view is shown in Figure 121 below.

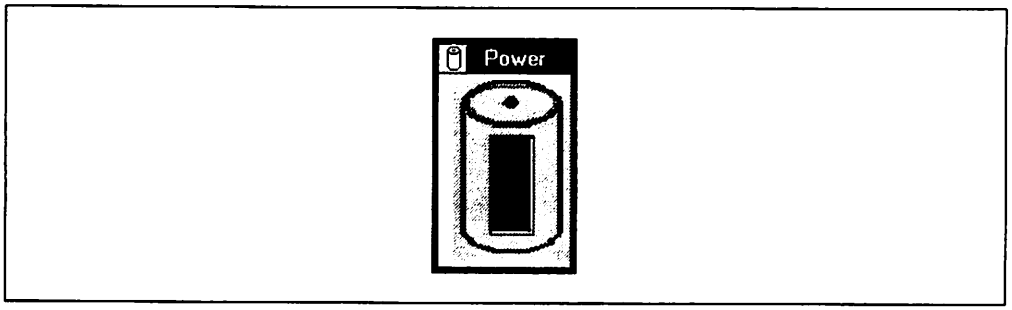


Figure 121. Power - Battery Status

If the battery status is **Unknown**, this view cannot be selected.

- Suspend

  The suspend mode allows you to save as much power as possible without turning the computer off. When **Suspend** is selected, the system dims the display and turns off devices that are not in use. If **Confirm on power status changes** is set in the Power Settings (refer to 11.1.4.2, "Power - Settings" on page 151), the following message is displayed.

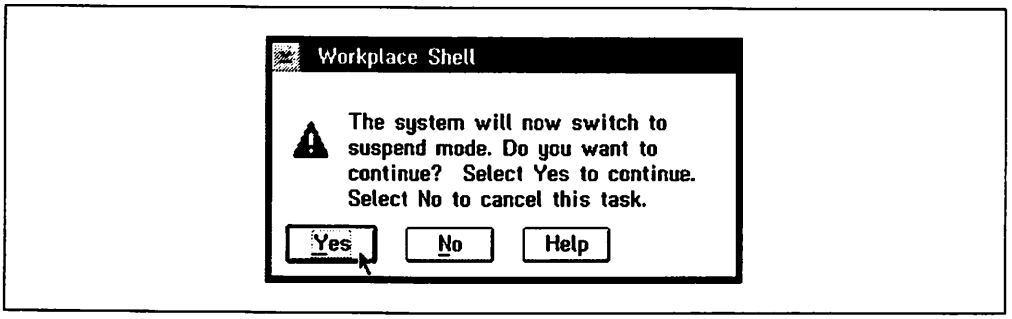


Figure 122. Power - Switch to Suspend Mode

To exit suspend mode and resume operation, you need to use a procedure that is different on different computers. Some systems have a **resume key**, other computers enter suspend mode when the computer lid is closed and exit suspend mode after the lid is opened. Refer to the documentation of your computer for information about its suspend mode features.

- Refresh

  Refresh can be set to On or Off, as shown in Figure 123.

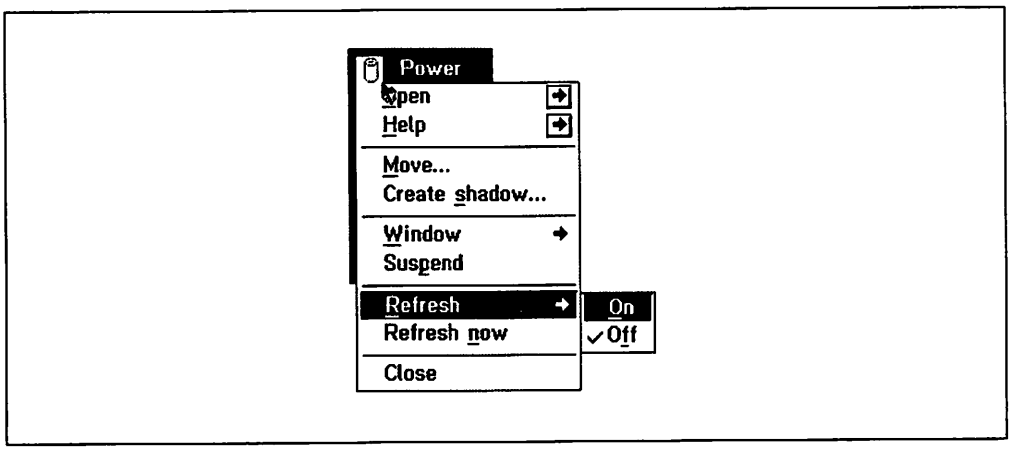


Figure 123. Power - Refresh

If **Refresh** is set to On, the system automatically updates the status window at intervals that can be set by the user.

- Refresh now

  This selection updates the power status immediately.

11.1.4.2 Power - Settings

A number of settings can be chosen by selecting **Open - Settings** from the status menu of the Power application. The first page of these settings is shown in Figure 124 on page 152.

Here, the power management can be turned on or off.

If **Confirm on power status changes** is checked, the user has to confirm the request to go to suspend mode. The message in Figure 122 on page 150 is shown when in this case. If **Confirm on power status changes** is not checked, the system immediately enters suspend mode upon request.

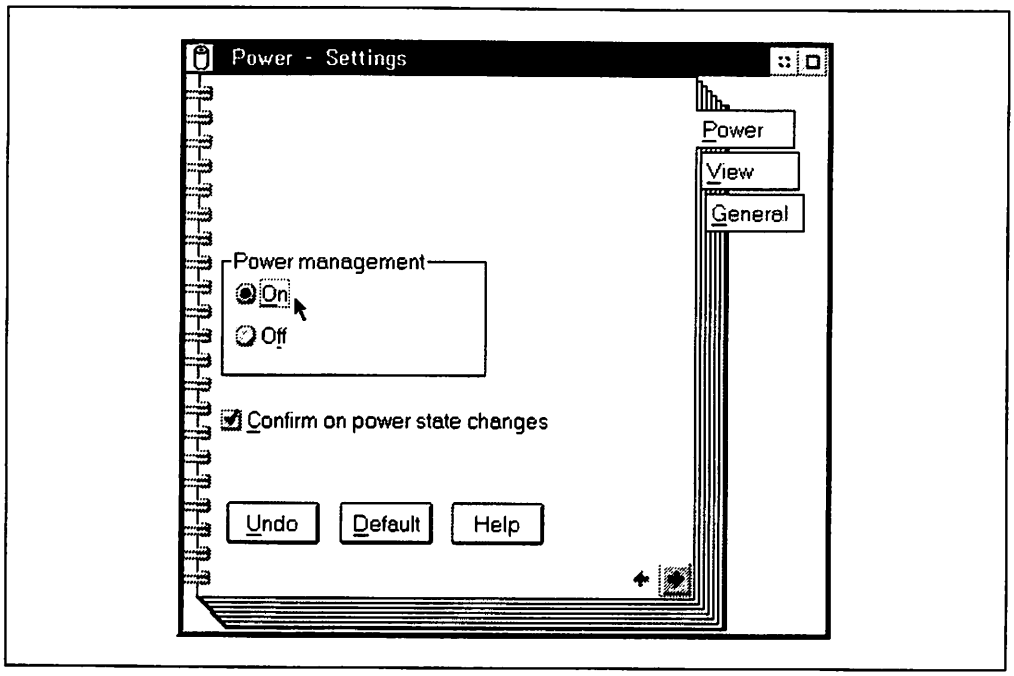


Figure 124. Power - Settings, Page 1

The second page of the power settings is shown in Figure 125.

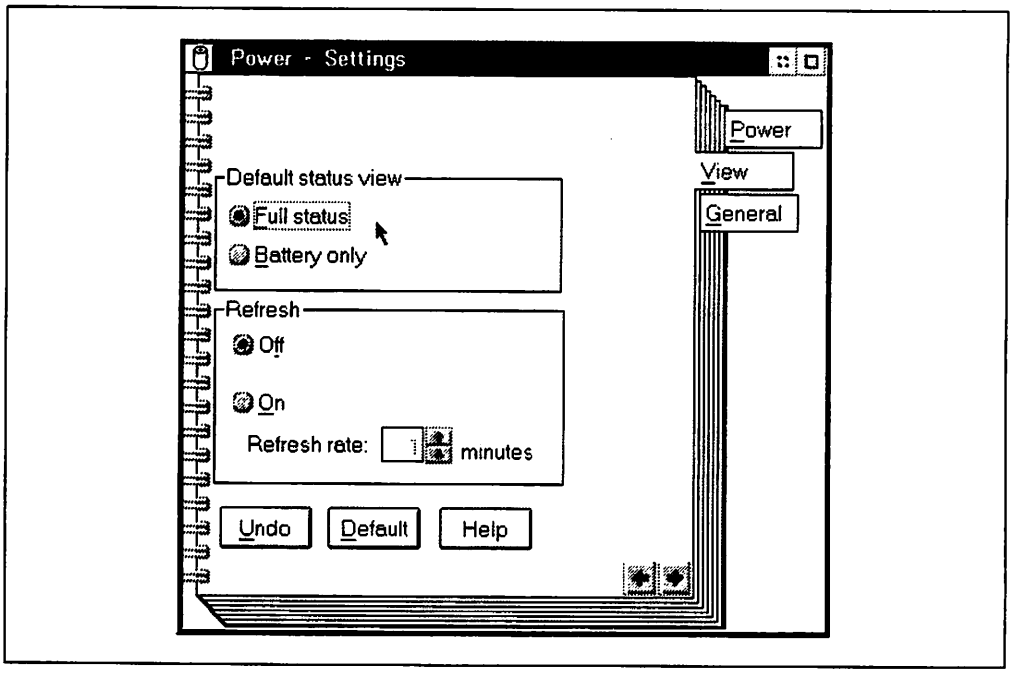


Figure 125. Power - Settings, Page 2

The default status view, **Full status** or **Battery status** can be selected here.

Automatic refresh can be set to On or Off. If refresh is set to On, the refresh interval can be set from 1 minute to 30 minutes using the spin button.

11.2 PCMCIA Support

This section describes the PC Card (personal computer card) as defined by the PCMCIA (Personal Computer Memory Card International Association) and the enhancements implemented in OS/2 2.1 to support such PC cards.

11.2.1 Introduction to PCMCIA and PC Card Technology

The Personal Computer Memory Card International Association (PCMCIA) is an organization of companies developing the personal computer card (PC card) standards. A PC Card is a small form-factor adapter for personal computers. PC Cards are about the size and shape of a credit card.

The PCMCIA is a non-profit organization that serves as a group for setting technical standards and as a trade association. The organization was formed to promote interchangeability of Integrated Circuit Cards (IC Cards) among a variety of computer and other electronic products. Members of the PCMCIA represent a wide range of computer industry segments, including hardware, software, semiconductor components, connectors, and computer peripherals.

11.2.1.1 Types of PC Cards

PCMCIA standards describe the physical requirements, electrical specifications, and software architecture for PC Cards. Three types of cards are described by the PCMCIA standards:

- Type I Card

  PCMCIA type I cards are 3.3 mm thick and may be used for various types of memory enhancements, including RAM, flash memory, one time programmable (OTP) memory, and electronically erasable programmable read only memory (EPROM).

- Type II Card

  PCMCIA type II cards are 5.0 mm thick and are mainly used for I/O features such as modems, LAN adapters, and host connectivity adapters.

- Type III Card

  PCMCIA type III cards are 10.5 mm thick and are mostly storage devices, such as miniature harddisk drives.

All three card types are 85.6 mm long and 54 mm wide and use the same 68-pin edge connector for attachment to the computer.

You can use PC Cards with suitable equipped laptops, notebooks, palmtops, tablets, and other portable computer systems, as well as some desktop computers. PC Cards are a convenient alternative to pocket adapters and docking stations.

11.2.1.2 Socket Services and Card Services

The key elements of the PCMCIA software architecture are Socket Services and Card Services. Socket Services is a BIOS level software interface that provides a way to access the PCMCIA sockets (slots) of a computer. Socket Services identifies how many sockets are in a computer system and detects the insertion and removal of a PC Card adapter while the system is powered on ("Hot Plugging"). Socket Services is part of the PCMCIA 2.0 specification and interfaces with Card Services.

Card Services is a software management interface that allows you to allocate the system resources (such as memory and interrupts) automatically, once the Socket Services detects that a PC Card has been added. Card Services also releases these resources when the PC Card has been removed. Furthermore, Card Services provides you with an interface to higher level software to load any needed hardware drivers.

11.2.1.3 Benefits of PC Card Adapter Technology

The combination of PC Card adapter hardware, Card Services software and Socket Services software provides a "plug-and-play" capability in the portable computing environment. Once the software has been installed, it is possible to add and remove PC Cards without powering off the system or opening the covers of the personal computer system unit. For example, you could insert a modem PC Card to access another computer system, download information into the portable computer's memory, remove the modem PC Card, replace it with a Flash PC Card, and store the downloaded information - all while your portable computer is still powered on.

As PC Card features are added and removed, you don't have to worry about what memory blocks, I/O ports, or interrupt levels are available. Card and Socket Services software configure the system automatically.

PC Card adapters provide you with the flexibility of adding the features you require after your base system has been purchased. One of the goals of PCMCIA is the interchangeability of PC Cards between portable computer systems. PCMCIA Version 2.0 slots can be added to ISA, EISA and Micro Channel personal computer systems. The same PC Card can operate in all these machine types with the appropriate Card and Socket Services software.

PCMCIA is a fast-moving, market-driven standards organization. It vigorously promotes worldwide acceptance and adoption of PC Card standards to ensure rapid development of the market. The association has two primary near-term objectives: to establish and maintain worldwide standards for PC Cards and to promote the standards through education. To reach these objectives, PCMCIA works closely with other trade organizations and standards groups including the Japanese Electronics Industry Development Association (JEIDA), the Electronic Industries Association (EIA), Joint Electronic Device Engineering Council (JEDEC) committees which standardize memory modules and packaging, and the International Standards Organization (ISO).

11.2.2 IBM Computers and Adapters with PCMCIA Support

At the date of publication, the IBM hardware products that conform to the PCMCIA standards are:

1. IBM Personal Computers with PCMCIA Support
   - IBM ThinkPad 720 and ThinkPad 720C

     These notebook computers can accommodate either one PCMCIA type III slot or up to two PCMCIA type I or type II adapters.
   - IBM ThinkPad 710T

     This pen-based computer comes in two models: the hard disk model and the ThinkPad file model. The hard disk model offers one PCMCIA type II slot, the ThinkPad file model ofeers three PCMCIA type II slots.
2. IBM PCMCIA Adapters

IBM is currently offering the following PC Cards that comply with the PCMCIA Type II standards:

- IBM Token-Ring 16/4 Credit Card Adapter
- IBM Credit Card Adapter for Ethernet
- IBM 3270 Emulation Credit Card Adapter
- IBM High Speed PCMCIA Data/Fax Modem
- IBM PCMCIA Data/Fax Modem

11.2.3 OS/2 2.1 PCMCIA Support

OS/2 2.1 provides support for PC Cards that conform to the PCMCIA standard (see 11.2.1, "Introduction to PCMCIA and PC Card Technology" on page 153).

If **PCMCIA Support** is selected during initial OS/2 2.1 installation or during Selective Install, the two device drivers for PCMCIA support are installed and the following two lines are added to the CONFIG.SYS:

```
DEVICE=C:\OS2\PCMCIA.SYS
DEVICE=C:\OS2\MDOS\VPCMCIA.SYS
```

These device drivers provide a PCMCIA Card Services interface.

There is no user-visible interface provided with the OS/2 2.1 PCMCIA support. Device drivers and applications to set up PCMCIA cards have to be provided by the manufacturer of the PC cards. These will include an interface that allows you to set up I/O addresses, RAM and ROM addresses, and IRQ resources used by a PC card, as required by the specific card you are using.

11.2.4 PCMCIA Compatibility Issues

The current PCMCIA standards, namely the PCMCIA PC Card Standard Specification release 2.01, the Socket Services Specification release 2.00 and the Card Services Specification release 2.00, have been published in November 1992. Some PC Cards were offered before this current standard was published; this led to PC Card manufacturers interpreting the standards differently. Consequently, some client device drivers may not take full advantage of Card Services and Socket Services, for example by not offering "Hot Plugging". Some manufacturers also bypassed the interfaces or wrote device drivers directly to applications.

It is therefore essential to test the PC, the PC Cards and the software for a given environment to ensure compatibility.

11.3 VGA Large Cursor Support on LCD Screens

The 32-bit VGA display driver includes a large cursor for use on LCD screens, in order to increase visibility.

LCD screens are typically used on laptop systems, such as the IBM ThinkPad series. Large cursors are provided for both the arrow pointer and the I-beam cursor.

The large cursor is not always configured automatically. It can be manually configured by using the following REXX command file:

```
/* */
call RxFuncAdd "SysIni", "RexxUtil", "SysIni"
call SysIni "USER", "PM_IBMVGA", "CURSOR_SIZE", "1",
say Result
exit
```

OS/2 2.1 must then be shut down and restarted in order for the new cursor to be displayed.

The small cursor can be reconfigured by using the following REXX command file (and shutting down and restarting the system):

```
/* */
call RxFuncAdd "SysIni", "RexxUtil", "SysIni"
call SysIni "USER", "PM_IBMVGA", "CURSOR_SIZE", "2",
say Result
exit
```

11.4 Trackpoint II Support

The ThinkPad 700 and 720 series of notebook computers include an in-keyboard pointing device, called the Trackpoint II.

This looks like a red joystick, but also includes complex algorithms to enable the Trackpoint II to be used in place of a mouse for graphical environments such as OS/2 2.1.

The ThinkPad keyboards include two extra buttons corresponding to the mouse buttons.

For more information on the Trackpoint II hardware, see the article "Trackpoint II: The In-Keyboard Pointing Device" in *IBM Personal Systems Technical Solutions, January 1993*, G325-5012-00.

Chapter 12. MMPM/2 - The OS/2 Multimedia Environment

The IBM Multimedia Presentation Manager/2 (MMPM/2) provides multimedia extensions to the OS/2 2.1 32-bit environment. This enables personal computers to run applications that combine sound, images, and video.

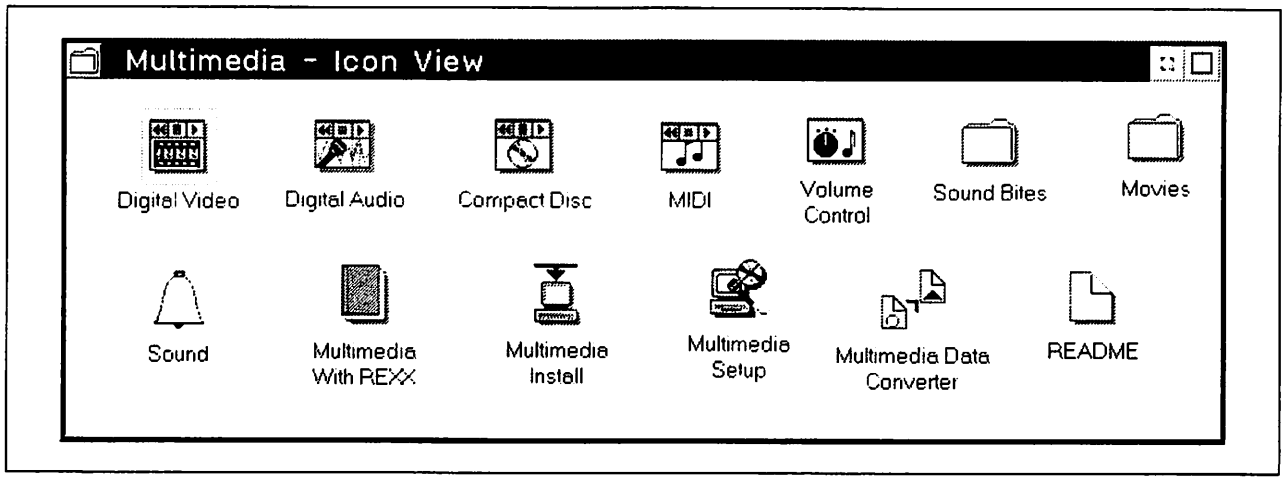


Figure 126. The Multimedia Presentation Manager/2 folder.

MMPM/2 1.1 is now packaged with OS/2 2.1, thus enhancing OS/2 Version 2 into a fully multimedia-enabled operating system platform.

12.1 Overview of MMPM/2

The key benefit of multimedia is an enhanced quality of information and communication. When audio, image, and video are combined with text and graphics, customers can access richer forms of information, and communication becomes more effective. MMPM/2 enables this increase in productivity by providing device control, streaming, synchronization, and multimedia object I/O support.

Using multimedia today also typically requires hardware components that previous personal computers do not have as standard - for example audio or video adapters, or CD-ROM drives. However, these can be easily added. Some personal computers - for example the PS/2 Ultimedia* models - include multimedia hardware as standard, and it is expected that this will become more common.

The MMPM/2 system supports the playback and recording of a variety of audio and video data types and formats. The MMPM/2 architecture enables the system to easily accommodate new functions, support new devices, and incorporate new data types and formats. The install program is designed with this growth in mind. As customers acquire new hardware or would like to add new functions, they can do so without re-installing the entire system. The open architecture for data standards support ensures that customer information assets retain their value regardless of format.

MMPM/2 also includes a set of player applications and utilities, and adds multimedia support to several existing system and application functions. Together, these enable the user to immediately take advantage of multimedia

capabilities, and to customize the system for their particular hardware and software needs. The user interface of the players, utilities, and the install program conforms to the 1991 Common User Access architecture.

MMPM/2's graphical install program is designed to be a "one-button" process if the user so desires. However, if customization is required, the user can select individual components, which can save disk space and available memory at run time. Context-sensitive helps are included throughout MMPM/2 to provide assistance with task completion, and to enable access to the most up-to-date product information.

MMPM/2 is provided in Universal English and the following eleven languages:

- French
- German
- Italian
- Spanish
- Norwegian
- Swedish
- Finnish
- Dutch
- Danish
- Japanese
- Korean

12.1.1 MMPM/2 Packaging and OS/2 2.1

MMPM/2 Version 1.1 is included and packaged with OS/2 2.1. This means that multimedia application users and developers do not have to worry about extra multimedia system support code, since OS/2 2.1 is already multimedia enabled.

Installation of MMPM/2 is still a separate procedure, but this is very straightforward. See 12.3, "MMPM/2 Installation" on page 166 for more details of the installation.

In the diskette shrinkwrap version of OS/2 2.1, MMPM/2 is provided on extra diskettes. In the CD-ROM shrinkwrap version, MMPM/2 is included on the CD-ROM.

MMPM/2 1.00 can still be purchased and installed on the base of OS/2 2.0, OS/2 2.00.1, or OS/2 2.0 with Service Pak XR06055 applied.

12.1.2 MMPM/2 Support in OS/2 2.1

MMPM/2 adds support for the following data types and formats to OS/2 2.1:

- Software motion video
  - Ultimotion*
  - Indeo**
- Audio
  - Digital audio (waveform)
    - PCM
    - ADPCM
  - MIDI
    - Type 0
    - Type 1

- Compact disc
  - CD-DA (Digital Audio)
  - CD-ROM
  - CD-ROM/XA
- Image
  - OS/2 bitmap
  - Windows DIB and RDIB
  - AVC* image
  - M-Motion* Image

Also, extensions are available to MMPM/2 for the following:

- Hardware-assisted video
  - M-Motion (analog video pass-through)
  - ActionMedia* II (digital video)

These extensions to MMPM/2 are shipped with the hardware adapters.

MMPM/2's extendable architecture enables new functions, devices, and multimedia data types and formats to be added as the technology advances. An example is the video functions for M-Motion and ActionMedia II that were added seamlessly to MMPM/2.

12.1.3 Software Motion Video

OS/2 2.1 supports an exciting new technology, software motion video. This is the ability to play video and synchronized audio without the need for special video hardware. MMPM/2 supports the AVI movie format using either the Ultimotion or the Indeo software motion video decompressor.

Ultimotion offers several significant advantages over other software video technologies currently available.

- Video adapters supported

  Ultimotion movies can be played on any OS/2-supported system that contains an SVGA or XGA video adapter (with 256 or 65,536 color capability). Ultimotion can also play back on VGA systems, although video quality is reduced when mapped to only 16 colors.

- Minimal system support

  This refers to playing a movie on a minimal system (25 MHz 386) directly from a compact disc (150 KB per second or less). Ultimotion supports movies that can play at 15 frames per second with a size of 320 x 240 pixels (one-quarter screen size on VGA or SVGA systems). This is *four times* the resolution of other typical software motion video technologies.

- Scalability

  Depending on the playback system, Ultimotion movies can be created with up to 30 frames per second, or at full screen size (640 x 480). All movies are created in a high-depth direct color space. If the playback system cannot support the movie as it was created, the Ultimotion technology will automatically scale the movie to match the system capabilities.

  If the processor in the playback machine cannot support the frame rate of the movie, Ultimotion will automatically scale down the video to match the

processor speed. The audio will play back at the speed it was originally created, and the video will stay synchronized with the audio.

The user can also scale the movie larger or smaller during playback, by changing the window size directly, or selecting options from menus.

The Ultimotion decompressor automatically plays the movie with 65,536, 256, or 16 colors, depending on the video adapter present in the system. With Ultimotion, there is no need to create separate movies with different numbers of colors — one movie covers all target playback systems.

12.1.4 Key OS/2 2.1 Features for Multimedia

MMPM/2 takes advantage of OS/2 features such as:

- Multitasking

Preemptive multitasking is essential for playing multiple data streams concurrently. With preemptive multitasking the operating system is in charge of effectively allocating the valuable resources of the processor. In contrast, cooperative multitasking, although effective for conventional applications, has a premise that all software developers will cooperate in adhering to a set of rules on how the processor should be utilized. This process is not well suited to multimedia environments, where guaranteed processor time is required for faithful playback of the data.

- Protected flat memory model

The "protected" memory allows the operating system to run applications in separate address spaces and prevents one application from corrupting the memory of another. This prevents system failures that require restarting the system because of ill-behaved applications.

The "flat" memory addressing architecture of OS/2 facilitates the handling of data objects many megabytes in size that are so pervasive in the multimedia environment. The ability to handle large objects as a single entity, rather than piecemeal, improves visual and file management performance.

- Fast threads

Fast threads are protect mode threads under OS/2 2.1 reserved for "time critical processing" such as those associated with streaming multimedia data objects through a system's memory.

12.2 MMPM/2 Contents

MMPM/2 consists of the following groups of components:

- **Device Drivers**

  These provide the device support for the specialized multimedia hardware.

- **Multimedia Subsystems**.

  These are the common operating system functional extensions needed for all multimedia applications.

- **Applets**

  These enable instant use of multimedia for simple tasks.

- **Multimedia Productivity Enhancements**

  Several system and application functions are now enabled with multimedia.

- **Utilities**

  These help with installation, setup and data conversion.

The functional content of MMPM/2 was determined by using Quality Functional Deployment or QFD. QFD is a process for understanding the needs of the customer and translating those desires into product characteristics.

IBM visited potential customers in the U.S., E.M.E.A. and A.P. to gather input related to multimedia systems. This data was used as the basis for deciding what functions to include in MMPM/2.

12.2.1 Device Drivers

Device drivers are shipped with MMPM/2 1.1 for the following multimedia hardware devices:

- Audio adapters
  - Sound Blaster**
    - Basic Sound Blaster
    - Sound Blaster Pro
    - Sound Blaster Pro, Microchannel version
    - Sound Blaster Pro 16
  - Pro Audio Spectrum 16**
  - IBM M-Audio*
- CD Audio (this runs on top of the appropriate OS/2 CD-ROM device driver)
- Video disc
  - Pioneer**

(The video disc device drivers are only shipped on the CD-ROM version of OS/2 2.1.)

In addition, OS/2 device drivers are shipped with many multimedia hardware adapters and components.

12.2.2 Multimedia Subsystems

Although there is a wide variety of multimedia hardware available, many of the tasks and design considerations are common across hardware devices. Examples include ensuring that audio or video data is reproduced smoothly on the screen or speaker, and what file formats are used for storing data.

Three multimedia subsystems are included as part of MMPM/2, to provide common services needed by the range of multimedia applications and devices. They are:

- **Media Control Interface**

  This provides a common, device-independent programming interface for multimedia applications. See 12.4.1, "Media Control Interface" on page 168 for more details.

- **Stream Programming Interface**

  This handles the key issue of ensuring that multimedia applications run smoothly in a multitasking environment - this is discussed in 12.4.2, "Stream Programming Interface" on page 168.

- **Multimedia I/O Services**

  These provide a common interface to accessing multimedia files and a common set of file formats, as described in 12.4.3, "Multimedia I/O Services" on page 169.

MMPM/2 also provides additional Presentation Manager controls that make it easier to create intuitive multimedia user interfaces. These are covered in 12.4.4, "Additional Multimedia Controls" on page 169.

12.2.3 Applets

A variety of simple multimedia applications are included with MMPM/2 1.1. They enable the user to immediately experience some of the potential of multimedia. The applets are:

- **Digital Video**

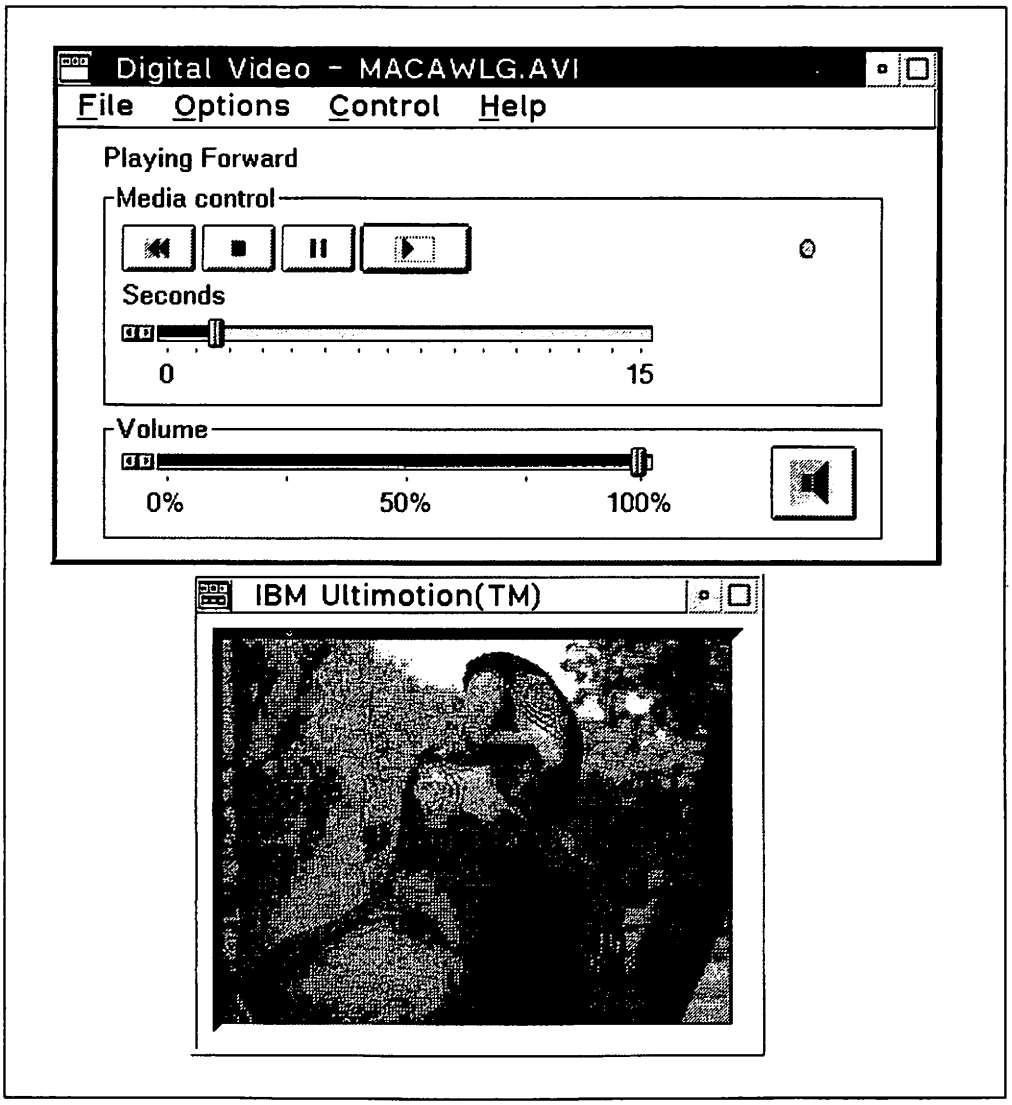


Figure 127. Digital Video Applet

This applet allows software-only playback of digital video files. The following video formats are supported:

- Ultimotion

– Indeo

Ultimotion movies can be created with the Video Workshop* product. Indeo movies can be created with other products that are currently on the market. For more information on software motion video and Ultimotion, see 12.1.3, "Software Motion Video" on page 159.

- **Digital Audio**

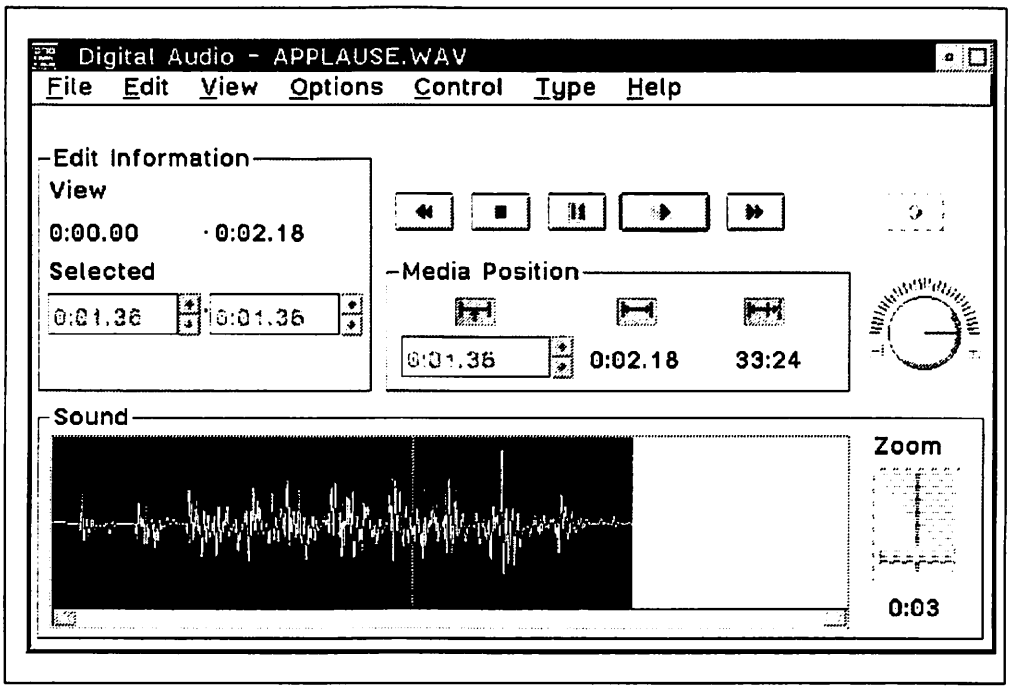


Figure 128. Digital Audio Applet

This applet allows digitally-recorded audio to be played back from a file through a supported audio adapter. It also provides recording support with enhanced editing capability. Special features include a graphical representation of the sound wave, cut/copy/paste, and various effects (such as mixing, fade in and out, and increasing and decreasing speed).

The user can select portions of the sound graph directly with the mouse, and then perform the editing functions on those portions. The graph can also be zoomed in or out for greater detail.

The Digital Audio applet can play PCM and ADPCM files, and records in the PCM format. The user can select the desired quality level for recording (voice, music, or high fidelity; mono or stereo; and 8 or 16 bit).

- **Compact Disc**

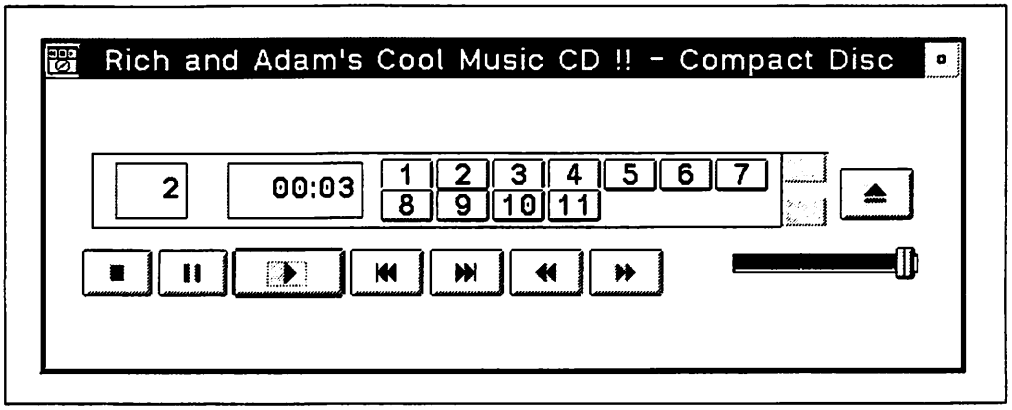


Figure 129. Compact Disc applet

This applet enables standard audio CDs to be played in CD-ROM drives. The interface for the player resembles what is commonly available on consumer CD players. It includes features such as track forward and backward buttons, scan buttons, direct track access, shuffle play, and customized titles for the discs themselves. This is all available in a size that does not occupy much screen space.

- **MIDI**

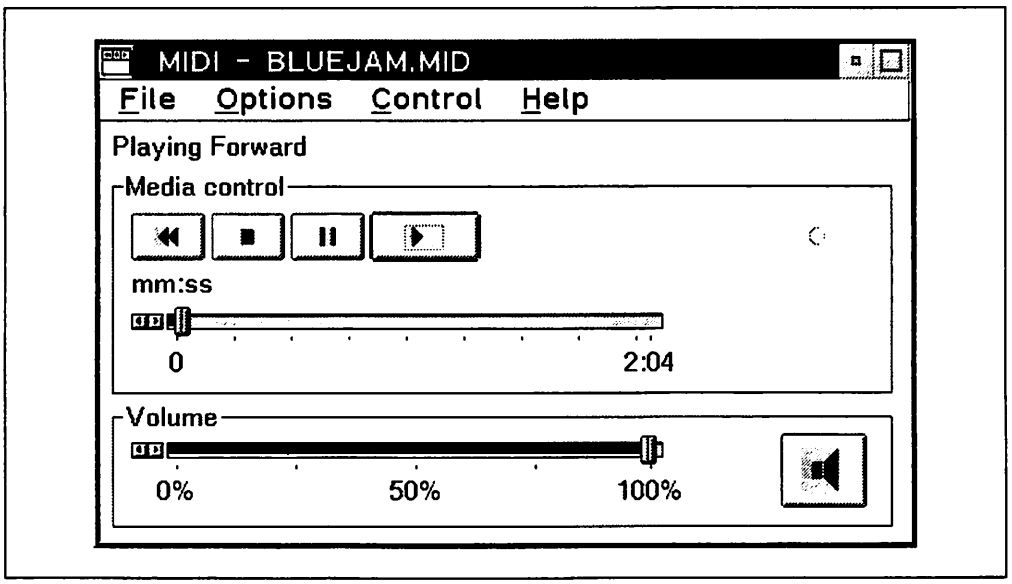


Figure 130. MIDI Applet

This applet enables synthesized music to be played through a supported audio adapter. MIDI types 0 and 1 are supported.

- **Volume Control**

  This applet provides a master volume control for all audio in the system, independent of the application that is generating the audio output. The Volume Control is shown in Figure 134 on page 169.

- **Sound Bites**

  This is a folder that contains a set of short digital audio and MIDI sequences that can be played with the appropriate players.

12.2.4 Multimedia Productivity Enhancements

Several system and application functions are now enabled with multimedia in OS/2 2.1. They include:

- System sounds
- REXX
- Spreadsheet applications

This enables immediate use of MMPM/2 functions, and allows the user to experiment with the concepts of enriched information through the use multimedia.

- **System Sounds**

  There are 13 system events that can now be associated with specific sounds. These events include:

  - System startup
  - System shutdown
  - Window open
  - Window close
  - Drag
  - Drop
  - Shred
  - Desktop lockup
  - Error
  - Warning
  - Information
  - Alarm
  - Printer error

  Audible feedback provides confirmation for the actions that are taking place, which can be especially useful in a multitasking system.

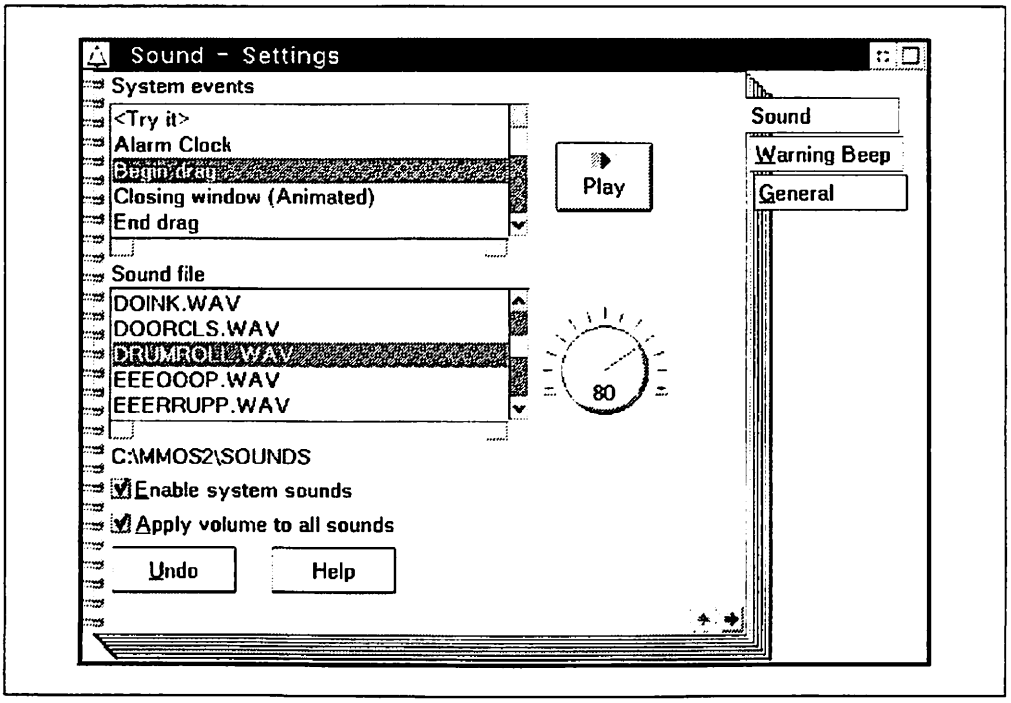


Figure 131. System Sounds Page in Sound Object

When MMPM/2 is installed, the existing Sound object (which can be found in the OS/2 System Setup folder) is enhanced. A notebook page is added to the Sound notebook. It contains a list of the system events and a list of the digital audio files that can be associated with the events. Any digital audio file can be associated with any event in the list.

A copy of the Sound object is also placed in the Multimedia folder for more convenient access.

- **REXX**

  The string interface to the Media Control Interface functions can now be accessed from REXX EXECs. This enables much of the power of MMPM/2 to be exploited in a non-PM environment.

- **Spreadsheet Applications**

  Audio-enabling spreadsheet macros are supplied with MMPM/2. They enable users of 1-2-3** and Excel** to record and play back audio annotations for spreadsheet cells. These macros work with the existing versions of these spreadsheet programs; no program updates are required.

  Users can now add comments about spreadsheet cells without having to type them in, which can make spreadsheet review easier and help clarify the meaning of individual cells. For example, if a complex formula is used on a cell, a verbal explanation could simply be attached to the cell. Using the features of these macros, users can record annotations, play them back, show or hide them, and delete them.

12.2.5 Utilities

Three utilities are provided with MMPM/2 1.1:

- **Multimedia Install**

  This provides easy graphical installation of MMPM/2. This is discussed in more detail in 12.3, "MMPM/2 Installation."

- **Multimedia Setup**

  This enables configuration of the multimedia devices installed on the system.

- **Multimedia Data Converter**

  This provides a tool to convert multimedia data between the different file formats. For example, you can convert a Windows DIB into an OS/2 bitmap format.

12.3 MMPM/2 Installation

Installation of MMPM/2 is separate from the installation of OS/2 2.1. In order to install MMPM/2, OS/2 2.1 must first be installed, and the system rebooted.

To install MMPM/2, use the MINSTALL program. This is located on either the first MMPM/2 diskette, or on the CD-ROM.

The installation program lets you customize your system, install a subsystem, or interrupt the installation if necessary. The subsystems you install depend on which devices you intend to use.

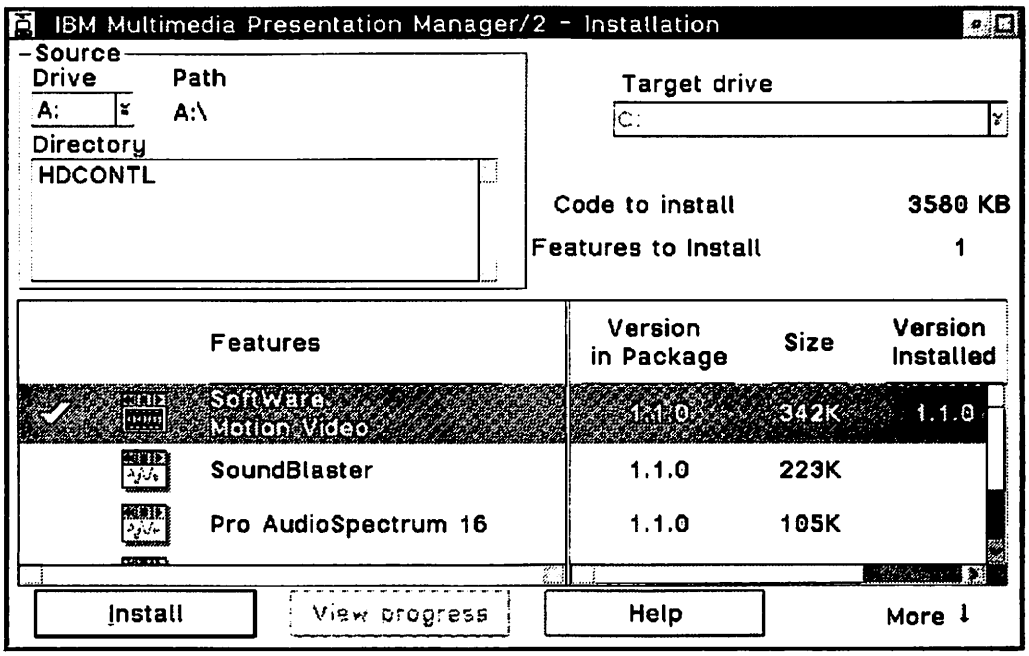


Figure 132. MMPM/2 Installation Main Panel

During installation, choices appear on your screen; you can either accept them or change them. If you want to accept the installation defaults, press Enter when each window appears. If you want to change a default choice, move the cursor to the information you want to change and select or enter a different choice. For more information on installing MMPM/2, refer to the *OS/2 Installation Guide* that comes with OS/2 2.1. This is also part of the OS/2 2.1 documentation-only package, which can be ordered separately as S61G-0905.

12.4 MMPM/2 Subsystems

As described above, there are three MMPM/2 subsystems which provide common system functionality needed by multimedia applications. In addition, new multimedia CUA controls are provided in order for multimedia applications to present a common look and feel to the user.

For more details on these subsystems, refer to the following publications:

- *The OS/2 Multimedia Advantage*, S41G-2923
- *MMPM/2 1.1 Application Programming Guide*, S71G-2221
- *MMPM/2 1.1 Subsystem Development Guide*, S71G-2223
- *CUA Guide to Multimedia User Interface Design*, S41G-2922
- *Multimedia Presentation Manager/2 Programming Using Digital Audio Examples*, GG24-3963
- *Multimedia Presentation Manager/2 Advanced Programming Techniques*, GG24-4042

12.4.1 Media Control Interface

The Media Control Interface provides a full 32-bit device-independent programming interface that is modeled after an audio and video home entertainment system.

This programming interface can either be a command message interface (a typical C procedure call, with parameters, messages and data structures), or a command string interface, which is probably easier to understand and use. Even though the programming interface is 32-bit, 16-bit applications can still be multimedia-enabled by using the string interface. There is no need to convert the applications to 32 bit.

The command strings are sent to the Media Device Manager (MDM) for parsing. For example, an application or a REXX EXEC could open a CD player. acquire and set device information, and play an entire CD using this set of command strings:

```
open cdaudio01 alias my_cd shareable wait
status my_cd media present wait
status my_cd mode wait
set my_cd time format milliseconds wait
seek my_cd to start wait
play my_cd notify wait
close my_cd wait
```

Figure 133. MMPM/2 - Example Set of MCI String Commands

The Media Control Interface gives applications greater flexibility, because application code does not have to be rewritten to support new devices as they are added to the system.

The Media Control Interface in MMPM/2 is a superset of the joint IBM and Microsoft specification for the Media Control Interface.

12.4.2 Stream Programming Interface

The Stream Programming Interface (SPI) ensures that video and audio playback is smooth. It does this by providing services for the implementation of data streaming and synchronization by media control drivers, eliminating the need for each driver to provide its own solution for these common multimedia requirements.

Pairs of stream handlers implement the transport of data from a source to a target device while the Synchronization and Streaming Manager (SSM) provides coordination and central management of data buffers and synchronization data.

Making SSM a centralized and architected feature of MMPM/2 guarantees a quality system solution to the synchronization of different data streams which has become especially significant in advanced multimedia applications requiring synchronized audio and video. The following stream handlers are provided with MMPM/2:

- Video
- Audio
- Multitrack
- MIDI mapper

- CD-DA audio
- CD-XA
- File system
- System memory

Additional stream handlers can be installed as required.

12.4.3 Multimedia I/O Services

The Multimedia I/O (MMIO) functions provide common access to a number of different multimedia file formats. They achieve this by enabling subsystem components, such as MMPM/2 media drivers and applications, to access and manipulate a variety of data objects, including digital video, digital audio, images, and graphics. These objects can be stored in a variety of file formats on a variety of storage systems.

Installable I/O Procedures (IOProcs) are provided to handle DOS files, memory files, and RIFF format files (both elements and bundles). Custom IOProcs can also be installed.

MMPM/2 provides the following I/O procedures:

- Video
  - AVI
- Audio
  - Wave
  - AVC audio
  - CLI VOC
  - MIDI
- Image and graphics
  - OS/2 2.0 bitmap
  - OS/2 1.3 bitmap
  - Windows DIB
  - Windows RDIB
  - AVC image
  - M-Motion image

12.4.4 Additional Multimedia Controls

MMPM/2 defines several new Presentation Manager window classes that lend a sophisticated multimedia quality to a user interface. These controls are the *graphical button*, *circular slider*, and the *secondary window*, as illustrated by the MMPM/2 volume control application.

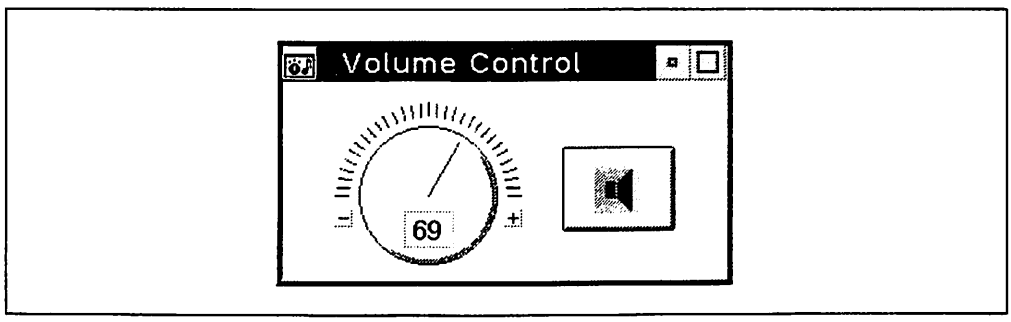


Figure 134. MMPM/2 Volume Control

The *graphical button* control allows replacement of conventional text-faced buttons with either static or animated graphics. The Mute button in the Volume Control applet is an example of a static graphic button. The Play button in the Digital Audio Player is an example of an animated button. While the object is being played, the triangle graphic moves from left to right across the face of the button.

Two-state buttons can be used as toggles. When pressed, they appear recessed into the screen. When pressed again, they return to their normal, raised appearance. A separate graphic can be defined for the raised and recessed states. The Mute button is an example of a two-state button.

Standard text-faced buttons can be replaced by graphical buttons without any change in source code. Only the dialog template needs to be modified. Implementing two-state and animated buttons requires a minimal investment in development time that is equivalent to implementation using the standard push button control.

The *circular slider* control presents an interesting alternative to the Workplace Shell slider control in that the physical appearance of the control is very similar to a knob typically found on a home stereo. This resemblance allows end users to transfer knowledge acquired from using real devices to multimedia applications. The circular slider can also provide a saving in screen real estate compared to a conventional slider. The code development complexity of using a circular slider is equivalent to using a standard slider or scroll bar.

The *secondary window* control represents a significant savings in development expense, because all application windows can be developed using a dialog template, thus eliminating the need to create, position, and size windows and controls. All of the MMPM/2 system applications utilize secondary windows. If the window is sized smaller than the actual dialog template, scroll bars can be automatically enabled to allow access to the entire dialog.

12.5 Benefits of MMPM/2 on OS/2 2.1

12.5.1 End User Requirements

When using multimedia applications users want to be able to play back and record high quality audio, images, and synchronized video concurrent with other computer tasks. OS/2's multitasking allows continual streaming of data without breakup, something cooperative multitasking cannot do. MMPM/2 was also architected to support the synchronization of data streams; therefore, no redesign was needed when video stream support was added.

When working with multiple applications a user wants the system to share multimedia resources amongst concurrently running applications. MMPM/2 provides for this by virtualizing devices so that multiple logical devices can exist for each physical device.

End users also need consistent control and extendable support of multimedia devices, especially for new device classes. MMPM/2 provides a complete set of window classes for the control icons used for multimedia applications. These window classes send the same messages to applications that current OS/2 CUA controls use (the circular slider for volume control, for example, versus the slider

bar for dialog box paging control). MMPM/2 is architected to allow the addition of new control tables, new media control devices, and new stream handlers to support new multimedia device classes, not just physical device drivers.

End users who have worked extensively with multimedia in the past have asked for ease of application and data migration from other platforms. MMPM/2 provides Multimedia I/O (MMIO) support for file type and format independence, and a data conversion utility that works with image and audio files.

12.5.2 Multimedia Application Requirements

Multimedia data objects are different from traditional computer data in two ways: they are very large, and they must be handled in real time. OS/2 provides priority-based preemptive multitasking and fast threads to reliably stream and synchronize these data types. Fast threads are protect mode threads under, OS/2 reserved for "time critical processing". They have improved dispatch latency over normal threads.

Also, because of their size, multimedia data objects are usually stored on the new higher capacity storage devices. The device driver support of CD-ROM, read/write optical drives and similar devices under the OS/2 file manager, is used by the streaming architecture of MMPM/2 and thereby provides the same real time streaming function to these devices as it does for hard files.

With the new OS/2 2.1 32-bit graphics engine, MMPM/2 is able to:

- Manipulate colors for image presentation
- Provide 640 x 480 and 1024 x 768 screen resolution with 256 and 64K colors
- Overlay images on top of each other
- Translate multimedia images when displayed
- Quickly display compressed bit maps allowing for large image databases to contain compressed images
- Change color palette when focus is switched from one window to another

12.5.3 Advantages of OS/2 for Multimedia

OS/2 offers several advantages for multimedia applications over the DOS and Windows environments:

- Preemptive multitasking instead of limited or cooperative multitasking.

  This allows for real-time control of multimedia devices allowing multiple data streams to be played concurrently with the ability to keep audio and video streams synchronized.

- 32-bit addressing support and flat memory model architecture.

  Large objects, such as the many megabytes in compressed digital video, means that vast amounts of data are associated with multimedia applications. These can be handled easier as a single piece rather then as 64KB segment chunks.

- Multi-threaded processes that share system resources, and provide responsive applications.

  Neither DOS nor Windows provide support for multi-threaded applications.

- Complete inter-process communication, including queueing, pipes and semaphores.
- A High Performance File System (HPFS) that is able to locate and move data objects faster than the File Allocation Table (FAT), the only format available under DOS.
- An extensive set of communications support through Extended Services and LAN requesters.

12.5.4 Functional Advantages of MMPM/2

MMPM/2 was implemented using an extendable architecture that enables the addition of:

- New functions and services
- New devices and device classes
- New data types and formats

The synchronizing and streaming manager portion of MMPM/2 provides superior support of multimedia streams:

- Multiple audio streams to one or more audio adapters
- Audio and video sequences with lip synch
- Synch control accessible from the application

In the Windows environment, synchronization requires low-level code with many callbacks, and multimedia applications typically need to have the input focus for this to function correctly.

The media device management, to support device sharing, allows better support for several multimedia applications running on one system. An amplifier mixer function acts as a master control of the hardware providing support for mixing, volume, balance, and treble of all audio streams. A Volume Control applet provides a single point of control for master volume and mute control of all audio on the system.

All multimedia applications developed under MMPM/2 have access to a comprehensive set of CUA multimedia data controls:

- Graphical and two-state buttons
- Animated buttons (play, for example)
- Circular sliders

MMPM/2 also allows for the playback and recording of memory objects at the Media Control Interface level, the migration of data objects through installable data handlers, and a user-level data conversion utility providing improved performance and investment protection of users' artwork, and audio and video content.

12.5.5 System Advantages of MMPM/2

MMPM/2 and its associated toolkit (MMPM Toolkit/2) provide extensive online and context-sensitive help for the developer and the end user. The multimedia applications developed using MMPM/2 have access to a rich set of application programming interface (API) multimedia functions such as:

- A consistent media control interface to:
  - Reduce application development time
  - Support today's hardware and adapt to tomorrow's technology advances
  - Allow device and stream management to be handled by the system
  - Ease programming by providing sample code for every major function in MMPM/2
- A standard data interchange format (MMIO) for the formats listed in 12.4.3, "Multimedia I/O Services" on page 169
- Multimedia enabling for 16-bit and 32-bit applications
- A common user interface that is extendable and provides a consistent look and feel across all multimedia applications and devices

The internal functions of MMPM/2 provide:

- Built-in high quality motion video support with four times the resolution of most other software-only video solutions
- A robust and extendable data streaming layer
- High priority threads for efficient data streaming
- Robust real-time synchronization functions
- Device virtualization and resource management (for example, DSP concurrency and load)
- Event synchronization through queue points and position advisers
- Support for communication systems

12.6 Applications for MMPM/2

12.6.1 Ultimedia Tools Series

The Ultimedia Tools Series is a collection of multimedia tools and applications intended to meet the needs of end users as well as casual and professional creators. The tools in this series let the user author multimedia presentations, create graphics and animation, edit audio and video, and manage multimedia objects. Three tools from IBM will be available for OS/2 2.1:

- **Ultimedia Builder/2**

  This is an authoring application that allows users to create online, interactive multimedia presentations. The user can directly manipulate icons to build the presentation, and include video, audio, image, and graphics. This is a superset of the AVC product.

- **Ultimedia Perfect Image/2**

  This application is specifically designed to allow users to categorize, convert, edit, and enhance images. Many different image types are supported, using MMPM/2's MMIO functions.

- **Ultimedia Workplace/2**

  This application organizes multimedia data in a highly visual manner that is consistent with the OS/2 Workplace Shell methodologies and interaction techniques.

Other companies have announced their intent to deliver OS/2 versions of their products in the near future.

12.6.2 Columbus: Encounter, Discovery, and Beyond

This is an educational application that includes text, graphics, images, digital audio, and video (from a videodisc). It is highly interactive, often presenting more than one data type at the same time (such as an image with digital audio that describes it).

While students can learn all about the life and times of Columbus, it is the exploration of the information itself that is its strong point. Students can navigate through the information freely, and take advantage of the powerful database query capabilities built into the application.

12.6.3 Illuminated Books and Manuscripts

This is also a multimedia educational application, geared toward the exploration of various literary works. When the student selects a work to examine (for example, the Declaration of Independence), it is presented on the screen. The student can then explore many aspects of the work via filters, presented as icons on the side.

For example, the student can select "Interpretations," and several passages of the work will become illuminated. The student can then select a particular passage, and the select one of several people to describe it. A video of the person discussing that passage then appears on the screen.

This application also contains a robust database search mechanism, and presents students with a variety of multimedia data types.

12.7 Summary of Advantages of MMPM/2 with OS/2 2.1

For multimedia applications, OS/2 2.1 has the technology to provide viable business solutions. Preemptive multitasking means that no application will unnecessarily consume the precious resources of the processor. Because this policy is enforced by the operating system, there is no need to rely on the good will of other application developers as in a cooperative multitasking system. This is especially important for multimedia applications where consistent throughput is essential in order to maintain the quality of a presentation.

Also essential to multimedia applications is the demand-paged virtual memory support in OS/2 2.1. Memory usage is optimized and the flat memory model makes programming straightforward. Memory is seen as a completely linear space up to 512MB. As with all previous versions of OS/2, OS/2 2.1 provides protected memory to guarantee a stable development and end user environment.

The addition of MMPM/2 creates a premier multimedia programming environment. All hardware devices are controlled through a full-feature, 32-bit, device-independent programming layer called the Media Control Interface.

Originally developed as a joint standard between IBM and Microsoft Corporation, the Media Control Interface greatly reduces the expense of adding multimedia capabilities to applications. In addition to insulating applications from the underlying hardware, MMPM/2 also shields applications from different data formats and provides a completely extensible architecture to allow the incorporation of new multimedia devices and multimedia data formats. A centralized synchronization and streaming manager provides dependable data transport and synchronization services.

With the solid foundation of OS/2, and the comprehensive multimedia services and applets provided by MMPM/2, OS/2 2.1 will help personalize the information revolution with an explosion of sights and sounds.

Chapter 13. Tuning and Performance Characteristics

This chapter focuses on issues related to OS/2 performance.

This chapter discusses what tuning and usage considerations are available to optimize system performance. This is presented from an administrator's perspective (tuning factors), as well as from an end user's perspective (usage factors).

This chapter also discusses what hardware factors (such as CPU or memory) can have an impact on performance.

Functional enhancements outlined in previous chapters are then summarized and their impacts on OS/2 performance are discussed.

13.1 Performance Considerations

Any time performance is discussed, the very nature of the discussion (and any conclusions to be drawn) is dependent on numerous factors. It is very important to put performance into perspective for *your* operating environment. Below are some of the factors which *may* have an impact on performance, although this may not hold true for your particular environment:

- Hardware factors
  - Processor type
  - Processor speed
  - Amount of memory
  - Type of disk
  - Attached devices
- Software factors
  - Type of applications
  - Network/host connections
- Usage factors
  - Multitasking
  - Workplace Shell utilization

The scenarios and operating environments presented in this chapter may not be the same as your exact environment and it is most important to keep this in perspective. Emphasis in this chapter is on the "concepts" of tuning and performance optimization, not on a particular test scenario and its respective results. Focus should remain on the "types" of tuning possible and the impacts that external influences such as amount of memory or types of applications have on your operating environment.

Throughout this chapter multiple views of performance tuning are presented. Some of these techniques are ones which you or your technical staff may perform, such as modifying the CONFIG.SYS file, while as others are internal to the operating system. This latter type of tuning is performed by IBM

programmers before the actual "build" of the operating system and its subsequent release to the public.

13.2 Optimizing Your System Configuration

The following section covers tuning from an *administrator's* perspective, focusing on modifying system parameters, setting up the Workplace Shell, and characteristics of application tuning. First, we will cover overall tuning considerations. This section is intended to summarize some of the *basic* concepts of tuning. 13.2.9, "Standard CONFIG.SYS Tuning Parameters" on page 183 and 13.2.8, "New CONFIG.SYS Tuning Parameters" on page 183 discuss many of the *specific* CONFIG.SYS statements and their parameters.

Application tuning is discussed from the standpoint of what an administrator or user can do (externally) to optimize an application's performance. Several Workplace Shell topics are discussed from a performance perspective, in an effort to provide multiple ways of setting up and utilizing the Workplace Shell. Lastly, effects of hardware or software upgrades and their tuning impacts are discussed.

Many of the items in this section may not increase performance through only a single change. However, regarding CONFIG.SYS parameters, if 4KB is saved here and 10KB is saved there (and so on) the system will benefit by having more memory available for applications and the system. There are big differences in some parameters and their impact on memory, whereas others are very minor (but may add up).

For more of the "user" activities and performance factors such as how to "work" OS/2 and the Workplace Shell to achieve optimal performance, see 13.3, "Optimizing Your Usage of OS/2" on page 190.

13.2.1 Tuning Considerations

There are a number of considerations that should be taken into consideration before modifying operating system configurations. This section will focus on the following considerations:

- Lazy-write
- Cache
- Workplace Shell
- Disk space
- CONFIG.SYS statements
- Applications

13.2.2 Why Tune?

First, the question of "Why tune?" should be answered. As the old country saying goes, "If it's not broken, don't fix it." This clearly holds true for altering configurations. Secondly, the question of "Who is going to tune it?" should be answered.

If you are not sure of the parameters and their settings, it may be wise to have someone who is technically knowledgeable look at your configuration. Lastly, have an agenda planned of what needs to be configured and an idea of what the

performance results should be. Make sure you have a baseline (where you are starting) and *realistic* goals (where you want to go) for your tuning efforts.

It is also very important to realize that an operating system is made up of interrelated components. Changes to one subsystem may have adverse effects on another subsystem. As an example, changing the caching size may keep more of your data in memory, but the application may not run with the reduced free memory. Most importantly, **change one parameter at a time!** If you change more than one, there is no sure way to know what change is producing what results.

13.2.3 When to Tune?

Most users will find that no tuning is required for OS/2 2.1. The default settings meet the needs of a wide range of user populations and requirements. This may not be true for all users and requirements. If you have special processing needs, OS/2 2.1 may need to be tuned to meet your needs. Tuning OS/2 may be required in other situations as well, some of which are:

- Hardware upgrades
- Software changes

Many of the settings for memory management such as cache and paging might be tuned if additional memory is added to your machine. It may be to your benefit to tune OS/2 if the application set on your system is changed.

Knowledge about OS/2 and how to tune it may also provide input for your future system expenditures. If 32-bit device drivers or applications are available to meet your needs, their performance in OS/2 will probably be superior to their 16-bit counterparts.

13.2.4 Lazy-Write Considerations

Lazy-write is one of the parameters available for the DISKCACHE and IFS= statements. Lazy-write describes the ability to buffer file writes on their way to the hard file and subsequently perform the actual write at a time when the disk controller is not busy with other disk activity. This delay is beneficial in terms of allowing the disk controller to perform higher priority actions first, and complete file writes when convenient.

Warning

However, there is an important implication in using lazy-write. With lazy-write active, any information "written" to disk (actually sent to the buffer) is actually kept in RAM, which is a *volatile* storage location. In the event of a system crash or power failure, anything in RAM is *lost*. For mission-critical or line-of-business applications, using write-through instead of lazy-write is strongly recommended, despite the performance gains possible with lazy-write. This is because using lazy-write would increase the risk of losing valuable data.

Lazy-write for FAT file systems is specified by the inclusion of the LW parameter on the DISKCACHE statement in the CONFIG.SYS file. This is shown below.

```
DISKCACHE=256,LW
```

The cache and lazy-write for FAT file systems *cannot* be dynamically set during a session. Any changes to the DISKCACHE statement will take effect the next time the system is shut down and rebooted.

Lazy-write for HPFS file systems is ON by default. It can be turned off by entering the CACHE command at a command prompt:

```
CACHE /LAZY:OFF
```

This can also be automated through a RUN= statement in the CONFIG.SYS file or through a batch file such as STARTUP.CMD.

Lazy-write can be turned on again by using the CACHE command. Since this command actually starts the CACHE program running, it should be run in a separate process using either the START or the DETACH command:

```
START CACHE /LAZY:ON
```

Turning lazy-write OFF will cause any data currently in the cache and all subsequent data writes to be written to disk.

The cache settings for HPFS volumes can be queried by entering CACHE at the command prompt. Values for the DiskIdle, MaxAge, BufferIdle, cache size, and lazy-write state are returned. These parameters are explained in 13.2.5, "Cache Considerations"

13.2.5 Cache Considerations

Cache is a facility which has the potential for dramatic performance gains in an operating environment. Basically, the ability to read information from memory as opposed to reading from an I/O device yields significant performance improvements. However, any memory set aside for the cache is memory which is taken away from the free memory available for applications. It is generally recommended that as much memory as possible should be dedicated to the cache after taking into account the normal operating requirements of your system.

This evaluation requires that you have a good understanding of the working set requirements of your system. For instance, in a machine with 24MB of RAM, and a calculated average working set of 10MB, up to 14MB could be devoted to the cache. 14MB of cache is a lot of space for your applications to write and read data, at speeds of nanoseconds rather than milliseconds. However, if the working environment changes, it is important to recalculate the optimal cache. Additionally, you may want to leave extra memory for applications that require more than their average working set for the initial loading of the application.

Note

The maximum size for the HPFS cache is 2MB. The maximum size for the FAT file system cache is 14MB.

The HPFS cache can be controlled through the parameters to the CACHE command. These are DISKIDLE, MAXAGE, BUFFERIDLE, cache size, and lazy-write. These allow you to customize the characteristics of the cache and lazy-write in terms of time and controller availability. The lazy-write parameter is discussed in 13.2.4, "Lazy-Write Considerations" on page 179.

The FAT cache has this information defined statically and it cannot be set by the user (apart from lazy-write).

The **MaxAge** parameter specifies the amount of time in milliseconds before data is transferred from the cache memory. The default is 5000 milliseconds. **DiskIdle** specifies the amount of time the disk controller must be inactive before receiving data from the cache. The default is 1000 milliseconds. **BufferIdle** specifies the amount of time in milliseconds that the cache buffer must be inactive before data in the cache buffer is written to disk. The default is 500 milliseconds. These parameters are also discussed in the *Online Command Reference*.

13.2.6 Workplace Shell Considerations

Although the Workplace Shell is the "user's" view of the operating system, there are a number of areas an administrator can modify or set ahead of time to increase performance and productivity for users. Performance in this case is not necessarily only in speed, but also in productivity and throughput. As an administrator, making the user's computing environment more efficient is a major benefit to productivity and job performance. Some of the areas to be discussed are:

- Workplace Shell CONFIG.SYS statements
- Startup folder
- Templates
- Applications

13.2.6.1 Workplace Shell CONFIG.SYS Statements

It is possible to create a consistent Workplace Shell startup for your users by using the SET RESTARTOBJECTS statement in the CONFIG.SYS file. The default is YES, which specifies that all objects which were active at shutdown will be restarted automatically. NO would indicate that no objects will be restarted. If the user environment makes use of the Startup folder, STARTUPFOLDERSONLY may be specified to restart only Startup Folder objects.

13.2.6.2 Startup Folder

The Startup folder is a useful facility for creating a standardized startup environment for the user. Administrators can customize the contents of the Startup folder to the exact needs of the user. Any objects placed in the Startup folder will open when the system is started. As a reminder, any active applications will also start when OS/2 is started. Use a combination of the Startup folder and the SET RESTARTOBJECTS parameters to achieve a *consistent* and clean startup. SET RESTARTOBJECTS specifies what objects are started when the Workplace Shell is started. If SET RESTARTOBJECTS=STARTUPFOLDERSONLY is specified, then only the contents of the Startup folder will start. Any applications which were running at shutdown will not automatically restart unless specified in the Startup folder. This can yield productivity savings for the user since there is a consistent and predictable startup of their machine.

13.2.6.3 Templates

The user environment can be customized for further productivity gains by utilizing templates. Creating this type of customized environment will also ease the administrative burden of supporting many users. OS/2 ships with default templates that can be customized for particular object types or application needs. The administrator can also create new templates for application specific needs.

An example of this is creating a text file with text headings and footings which are specific to a particular division in your company. Once this file is created with only the header and footer information, the Template radio button can be selected from the General tab of the text file's Settings Notebook. New copies of the customized text file can now be dragged off the original and all the customizations will appear by default.

13.2.6.4 Applications

Running applications in a full screen session will almost always yield improved performance over windowed sessions. This difference is primarily due to the translation OS/2 has to perform when writing the application into a window on the desktop. WIN-OS/2 3.1 has a different palette than Presentation Manager, which forces a translation to occur when running a seamless WIN-OS/2 3.1 session. Avoiding this translation should improve performance.

13.2.7 Disk Space Considerations

Disk space is a very important asset and there are several ways to save disk space if your environment is limited. Once familiar with the Workplace Shell and its facilities, many of the files can be deleted to save disk space. Examples of this are the tutorial and one of the text editors, since OS/2 ships with two editors. The WIN-OS/2 and DOS environments can be deleted if support for these environments is not needed, saving approximately 8MB and 7MB respectively (OS/2 2.0 capacities).

The mini-applications and games that ship with OS/2 consume a considerable amount of the default installation's disk requirements. If the games and productivity applications are not needed, over 5MB of disk space can be recovered. The applications and games are located in the \OS2\APPS subdirectory. It is important to note that when deleting the applications or the games, be sure to delete their respective dynamic link library (DLL) files from the \OS2\APPS\DLL subdirectory.

An alternative to deleting files is to plan ahead during the installation. Much disk space can be saved by using the selective install process and selecting only those facilities which are necessary for your processing needs. This is beneficial if you are familiar with OS/2. If you are not familiar with OS/2, we suggest that you install all of the operating system and explore all of the features which come standard and after you have determined what is useful for you, selectively delete the other information.

Disk space is important for numerous reasons. Space for applications and data are obvious reasons to have abundant disk space. Additionally, disk space is important if you are operating in a memory constrained environment. Making use of OS/2's virtual memory capability can yield great benefits, although the paging facility can require significant disk space for the SWAPPER.DAT file.

Warning!

Caution should be used when deleting files from the hard disk. If you are not sure what a particular file represents, **do not delete it!**

13.2.8 New CONFIG.SYS Tuning Parameters

The MEMMAN statement has a new parameter under OS/2 2.1. The new parameter is COMMIT and it is *optional* and is not present by default. The parameter must be manually added to the CONFIG.SYS file if overcommit accounting is desired such as:

```
MEMMAN=SWAP,COMMIT
```

This is used in conjunction with the MINFREE parameter of the SWAPPATH statement. Please refer to 13.5.3, "Paging Management Changes" on page 199 for more information on these parameters and an explanation of overcommit accounting.

13.2.9 Standard CONFIG.SYS Tuning Parameters

The CONFIG.SYS statements for the OS/2 2.1 release are similar to those in previous versions of OS/2 with a few notable exceptions. Below is a brief explanation of some of the statements in the CONFIG.SYS file which may have a direct impact on system performance. Further descriptions can be found in the *Online Command Reference*.

As the size of the CONFIG.SYS statement would indicate, there are quite a few parameters to contend with. In an effort to isolate those that have a direct impact on performance, the section below outlines statements that either add device drivers (and add to what is loaded into the working set) or specify how specific features of the operating system work. The statements that are discussed below are:

- IFS statements
- SET RUNWORKPLACE statement
- LIBPATH statement
- PRIORITY_DISK_IO statement
- DEVICE statements
- DISKCACHE statement
- MAXWAIT statement
- MEMMAN statement
- THREADS statement
- SET DELDIR statement

There are additional considerations which the "administrator" should be concerned with. Many of these are related to the Workplace Shell and how to optimize the Workplace Shell settings for the best performance. As mentioned numerous times above, these parameters are very "environment" specific. It is important to know the type of environment you, or your users, are working with before extensive modifications are made to the operating system.

13.2.9.1 IFS Statements

The IFS statements in the CONFIG.SYS file are for Installable File Systems, such as HPFS or CDROM. Since HPFS is the most common, we will use it for our example. The important aspects of the IFS statement are CACHE and Threshold size (CRECL). A sample IFS statement for HPFS may look like the following with OS/2 installed on the "D" partition:

```
IFS=D:\OS2\HPFS.IFS /CACHE:256 /CRECL:4 /AUTOCHECK:CDE
```

The /AUTOCHECK parameter specifies which hard files (formatted as HPFS) get checked (CHKDSK) on startup. AUTOCHECK and CHKDSK are very important in the OS/2 environment. OS/2 makes extensive use of extended attributes (EAs), which are fields set aside for each file for information such as file type and other OS/2 related information. If a disk stops incorrectly, such as when OS/2 is powered off rather than shut down, the extended attributes may be corrupted or damaged. The CHKDSK will notify you of problems and provide rudimentary means of recovering EAs.

The /CACHE parameter has very important and tangible impacts on performance. It is normally advisable to make the cache size as large as possible, once the operating environment is taken into consideration. To optimize how much memory to delegate to the cache, first calculate the average working set size for your environment. Ideally, do this over a day, or over "some" logical time frame for your work habits. Subtract this working set number from total physical memory, and the result could be delegated for the cache. Note that if your processing requirements increase and the amount of memory required increases, you will want to reduce the cache size to allow room for additional processing. Remember that the cache size is limited to 2MB for HPFS and 14MB for FAT.

The /CRECL parameter works integrally with the /CACHE parameter and specifies the threshold record size of reads into the cache. The default is 4KB and must be specified in 2KB multiples. The maximum is 64KB and the minimum is 2KB. In the above statement, the cache will read a maximum record size of 4KB into the cache. If the read is more than 4KB the cache will be bypassed and the application would receive data without having it cached. You may want to increase this depending on the type of reads your application performs.

Larger size records can be read into the cache by increasing the threshold. The threshold size cannot exceed one-fourth of the total cache size and if specified too large will be ignored. If the threshold on a system with 2MB cache specified was set to 1MB (one-half), the threshold would be ignored.

Note

If HPFS is the sole file system, be sure to remove (or REM) the DISKCACHE statement from the CONFIG.SYS file. The presence of the DISKCACHE statement will still allocate memory for the FAT file system cache, even without the FAT file system being used.

13.2.9.2 SET RUNWORKPLACE Statement

The SET RUNWORKPLACE statement refers to the program that is desired to be run as the user interface. The default is PMSHELL.EXE, which is the Workplace Shell. There are certain environments which may not take advantage of the Workplace Shell and could benefit by running an alternative "shell". For instance, if your environment uses OS/2 primarily for its system integrity and runs only one full screen application, it may be advantageous to run an alternative shell or that sole program as the user interface.

Because of the Workplace Shell's tremendous functionality and versatility, it is also a large application with large working set requirements. The Workplace Shell is a Presentation Manager (PM) application and any other PM application could be used in place of it. In this case the application would be the only application running. Instead of the Workplace Shell or a PM application, you could run an OS/2 command prompt as the "shell". This would allow you to start other processes and still perform multitasking without the overhead of the Workplace Shell. If OS/2 is installed on drive "D" the SET RUNWORKPLACE statement would look like the following:

```
SET RUNWORKPLACE=D:\OS2\CMD.EXE
```

Note that there will be limitations on your operating environment depending on the type of application you start as the user interface.

13.2.9.3 LIBPATH Statement

The LIBPATH statement sets the path where OS/2 will look for the Dynamic Link Libraries (DLLs) for system and application usage. There may be a slight performance impact from this variable, depending on the length of the statement and the location of frequently accessed DLL subdirectories. If your applications all have individual storage locations for their DLLs, and each application has added its DLL subdirectory to the LIBPATH statement, a large amount of querying will have to be done in order for the application to find the associated DLLs.

A file management scheme may help in grouping DLL files together, while still maintaining a version management system. If DLLs are grouped together, only one entry is needed in the LIBPATH statement. An alternative for some applications is to specify a period followed by a semicolon as the first entry in the LIBPATH such as:

```
LIBPATH=.;D:\OS2\DLL:....
```

This tells the operating system to search the current directory for DLLs first, thereby avoiding extensive queries to the file system.

13.2.9.4 PRIORITY_DISK_IO Statement

The PRIORITY_DISK_IO statement specifies whether or not the application in the foreground (active) should receive priority boost for file input and output. The default is PRIORITY_DISK_IO=YES. Normally, this is desired and will provide the best performance throughput. However, there may be some occasions where the background applications are the time-critical applications which would be adversely affected by another application receiving file input/output priority boosts.

13.2.9.5 DEVICE Statement

DEVICE= statements are specified in the CONFIG.SYS for device drivers which are loaded at system startup. These device drivers load a portion of their memory object into the resident memory area (fixed) and therefore increase the working set. A sample DEVICE= statement for a fictitious plotter is shown below:

```
DEVICE=D:\DEVICES\PLOTTER.SYS
```

Device drivers for DOS sessions can also be specified using the DOS_DEVICE setting in the Settings notebook.

Device drivers are somewhat of a necessary evil, in that most peripheral devices or operating system functions require device drivers. This gives functionality, at the expense of working set memory.

However, since some devices are used rather infrequently, it may be beneficial to keep several remarked DEVICE statements in the CONFIG.SYS file. The REM can be taken out by editing the CONFIG.SYS file whenever the device is needed. Since the CONFIG.SYS file sets the system environment at startup, it is necessary for a reboot with most changes to the CONFIG.SYS.

13.2.9.6 DISKCACHE Statement

DISKCACHE specifies the amount of memory devoted to the FAT file system cache (in KB). As with the HPFS cache, this parameter can have a dramatic impact on performance. A sample DISKCACHE statement is presented below:

```
DISKCACHE=128,LW,32,AC:C
```

The first parameter specifies the size of the DISKCACHE in KB. This is the amount of space taken out of physical memory to store information read in from disk. The second parameter specifies lazy-write, and its presence in the statement indicates lazy-write is enabled. Lazy-write refers to the facility which allows an application to perform a write to the file system and the write actually goes into a buffer before being written to disk. Please refer to 13.2.4, "Lazy-Write Considerations" on page 179 for additional details. Lazy-write uses an algorithm to determine when the actual write to disk is performed. It either writes it to disk when the disk controller has been inactive for a period of time, or when a certain proportion of the buffers have been written to and are thus dirty.

The third parameter refers to the threshold size, in sectors, for file reads into the diskcache. This is similar to the HPFS threshold, except for its specification in sectors. The last parameter refers to AUTOCHECK which, when specified, will perform a CHKDSK on the specified drives at system initialization. AUTOCHECK is useful in terms of its ability to indicate problems with partitions (for example, bad sectors), but its processing time will increase the overall amount of time to boot up the computer. Please refer to the *Online Command Reference* for additional information.

13.2.9.7 MAXWAIT Statement

The value for the MAXWAIT statement specifies the length of time, in seconds, a process waits before the system assigns it a higher priority. This parameter should be changed only with in-depth knowledge of the operating environment and potential impacts on priority changes. Three (3) seconds is the default. If there are multiple time-critical applications running, lowering MAXWAIT from 3

seconds would ensure that each application would receive priority boosts more frequently. This also has potential drawbacks, since lowering the MAXWAIT effectively reduces the amount of time the processor spends executing for each application.

13.2.9.8 MEMMAN Statement

As mentioned in 13.5.3, "Paging Management Changes" on page 199, the new parameter on the MEMMAN statement is COMMIT. This is an optional parameter, and the default is NO COMMIT. The parameter would be specified as follows:

```
MEMMAN=SWAP,PROTECT,COMMIT
```

When nothing is specified for COMMIT, the SWAPPER.DAT management is the same as in previous releases of OS/2. With COMMIT specified on the MEMMAN statement, the MINFREE value specified on the SWAPPATH statement now indicates the amount of disk space which will remain free on the partition. Previously, MINFREE specified the point at which the operating system would post a pop-up dialog box warning that disk space is running low. The SWAP/NOSWAP and PROTECT parameters retain their original meaning.

13.2.9.9 THREADS Statement

The THREADS statement specifies the maximum amount of threads which can be created by OS/2 for itself and applications. The maximum value is 4096, and the default is 256. A rule of thumb is to count 3 threads per application and add 60 extra. Each thread requires memory to maintain it, and this may be a factor in memory constrained environments. If this is the case, reduce the value for THREADS to 128 (the minimum).

13.2.9.10 SET DELDIR Statement

SET DELDIR refers to an area on each partition where space can be set aside to collect data that has been deleted from the system. For example, if a data object is dropped on the shredder, the object would actually be copied into the path specified by the SET DELDIR statement. The statement is in the default CONFIG.SYS, although it is commented out (REM SET DELDIR). To activate, remove the REM from the statement below, and modify the size if necessary (size is in KB).

```
REM SET DELDIR=C:\DELETE,512;D:\DELETE,512;
```

Note that the DELDIR facility operates on a per-partition basis. Any objects deleted from the "C" drive will only be saved into the C:\DELETE subdirectory. If you have multiple partitions, you will want to specify multiple delete directories.

The delete directory is a temporary storage location for files which are shredded, and it is only as large as specified on the SET DELDIR= statement. The delete directory facility uses a First-In-First-Out (FIFO) method to determine what is discarded from the delete directory when the maximum size has been reached. For example, if 512KB is specified, and three 150KB files have been erased, the next 150KB file to be deleted will cause the delete directory to discard the first 150KB file to make room for the recent deletion. This is a wraparound characteristic.

The last file is always saved, even if it is larger than the maximum size of the delete directory.

Temporary files created and then erased, for example by a compiler, will also be saved automatically, and this could flush out other, more important files from the buffer.

To recover files that may have been inadvertently erased, the command is UNDELETE and is issued from a command prompt. This will run through the delete directory and prompt for undeletion of each file currently in the delete directory. When a file is deleted, it is copied into the delete directory and a control file is created. The control file contains information necessary for undeletion, and for FIFO accounting of the file space. All files in the delete directory have system and hidden attributes (+SH).

As for the performance impacts, there are a couple of minor implications for using the delete directory facility. There is overhead associated with the safeguard of having a temporary "trash can". For example, the file which was dragged to the shredder is copied to the delete subdirectory, rather than erased, which takes additional processing. Also, there is a performance hit in terms of disk space utilization. Depending on the size specified on the SET DELDIR statement, disk space could possibly more efficiently used. Lastly, there is the possibility for a performance improvement in terms of productivity. Having an UNDELETE capability could save valuable time for file recovery.

13.2.10 DOS and WIN-OS/2 Application Environments

There are some general considerations for DOS and WIN-OS/2 sessions that may help improve performance. Two of the main items are memory allocation for DOS sessions and the session mode. Session mode refers to running the application (DOS or WIN-OS/2) in full screen or seamless mode. Virtually all applications run in full screen will have better throughput than running the application in a seamless session. This must be weighed against the convenience of having DOS and WIN-OS/2 applications running side-by-side on the Workplace Shell.

Memory allocation is very important in DOS and WIN-OS/2 (MVDM) sessions and has a direct impact on system performance. The default memory allocation for MVDM sessions is 2MB. Many applications cannot make use of 2MB (let alone 1MB). Unless your application specifically requires 2MB or more of DPMI memory, it is beneficial to reduce the DPMI_MEMORY_LIMIT parameter in the DOS settings of the application's Settings notebook. More information is available about the DOS settings in *OS/2 Version 2.0 - Volume 2: DOS and Windows Environment*, GG24-3731 and in Chapter 6, "MVDM Enhancements" on page 65.

13.2.10.1 Changing MVDM Settings

Many of the Virtual DOS settings can be set by administrators provided they have knowledge of the user environment. Some of the important MVDM settings which affect performance are:

- DOS_BACKGROUND_EXECUTION
- DPMI_MEMORY_LIMIT
- EMS_MEMORY_LIMIT
- XMS_MEMORY_LIMIT
- INT_DURING_IO
- DOS_RMSIZE

- VIDEO_RETRACE_EMULATION
- Full Screen or Windowed

These settings are changeable through the Settings notebook for each MVDM application.

DOS_BACKGROUND_EXECUTION is one of the important DOS settings. Most DOS applications perform polling and interrupt processing even when the application appears idle (foreground or background). When you switch back to OS/2 from a DOS word processor, the DOS application is still using CPU cycles. If the word processor is not running macros or actually performing any work, it makes sense to turn the DOS_BACKGROUND_EXECUTION setting to *off*. Other applications will then receive larger shares of the CPU for their processing needs.

If your application needs to process interrupts all the time, as many communication packages do, you should not changes this setting to OFF. The default for DOS_BACKGROUND_EXECUTION is ON. These can be set ahead of time by the administrator to minimize the interaction of the user with the system settings.

XMS_MEMORY_LIMIT, **EMS_MEMORY_LIMIT** and **DPMI_MEMORY_LIMIT** also can have an important impact on performance. The default settings for the DPMI memory limit is 64MB and for EMS and XMS memory limits are each 2MB. If an application cannot take advantage of these types of memory, reduce these settings to zero. On the other hand, a single WIN-OS/2 3.1 session with multiple applications active may require more memory to be set. It is best to test the various memory settings for each application scenario before setting extremely high amounts in the three MEMORY_LIMIT categories.

A setting of 2MB for EMS_MEMORY_LIMIT requires OS/2 to allocate this amount when the DOS session is started. These 2MB allocations can quickly add up if multiple DOS applications are started. If the application cannot use this type of memory the overhead on the system is unnecessary and may cause excessive swapping.

INT_DURING_IO is one of the new DOS settings with the Service Pak release. This setting is set to OFF by default. INT_DURING_IO creates a second thread for the application to use for interrupt handling when the primary thread is busy with I/O operations. This is very useful in multimedia applications, MSCDEX applications, and many games. INT_DURING_IO does create extra overhead on the system for processing and memory requirements and can cause degradation of performance for other applications. For this reason, unless your application is interrupt sensitive, leave this setting OFF. For additional information on INT_DURING_IO and other DOS parameters, please refer to Chapter 6, "MVDM Enhancements" on page 65.

DOS_RMSIZE refers to the amount of program memory to be allocated for MVDM sessions. Many DOS sessions do not use the full 640KB of memory available to DOS. DOS_RMSIZE could be lowered for these sessions to reduce the overhead associated with allocating memory for each MVDM session. An example of this type of application might be a DOS text editor. Some video adapters may be able to make use of this program memory if the application does not need it. Settings are in 16KB increments from 128KB to 640KB.

VIDEO_RETRACE_EMULATION is a facility provided by OS/2 to MVDM sessions that emulates screen retraces for the application. Some applications have timing difficulties with this emulation ON. Setting this to OFF will allow the application to control the video retrace. Some applications have timer ticks based upon video retrace. When OS/2 is doing the retracing, the applications will have unpredictable speeds and results. It is important to note that switching between OS/2 and applications with VIDEO_RETRACE_EMULATION set to OFF may encounter palette inconsistencies. Therefore, only switch VIDEO_RETRACE_EMULATION to OFF if your application is having difficulties in the video area.

13.3 Optimizing Your Usage of OS/2

This section is intended for users of OS/2 who may not have in-depth experience with computer systems but have been using OS/2 to some level and have at least been through the *Online Tutorial* provided with OS/2. Some of the information can also be found in the "Start Here" application which is loaded on the Workplace Shell desktop at installation time. If you experience difficulty operating OS/2 it is important to seek assistance before going any further. OS/2 and its graphical user interface are intended to be intuitive and easy to use but this will vary depending on your own conventions and customs.

13.3.1 Level-Set Performance Expectations

It is important to have a realistic approach to gauging the performance of your system. The capabilities of OS/2 are enormous compared to single tasking operating systems. However, many will immediately begin to run as many applications at once and begin to realize their system's performance is not as good as it was with only one application active. Matching the processing power to the tasks a system is asked to perform is very important. The more you want to do, the more processing power you need, regardless of the operating system. As you utilize and make the most of OS/2, keep in mind *what* work you are trying to do and *how* you are trying to do it.

13.3.2 Day-to-Day Usage Considerations

Some of the more obvious impacts on user performance are:

- Number of active applications
- Types of active applications

There are also some less obvious impacts on performance. Some of these are:

- Number of icons on the desktop
- Number of open folders
- Number of active bitmaps

It is important to think about what OS/2 is doing in the background to accomplish all the tasks you want to do and to maintain the look and feel you want. It is great to have an object-oriented user interface for look and feel, ease of use, and other advantages. The drawback with object-oriented interfaces is the resources required to process all the objects and object-related information such as folder size, position, color, and content. As a folder is opened the amount of memory needed to process the information increases and places a burden on other applications or tasks which are all competing for memory and processor resources.

13.3.2.1 Folder Considerations

It is helpful to close unused folders and applications that are not in use. We also suggest minimizing unused folders in order to improve responsiveness. Each folder requires an amount of memory to maintain information about the folder, such as folder icon, folder type and others. The same holds true for objects within folders. As a folder is opened, the operating system queries the file system (using DOSFindFirst and DOSFindNext) to determine what the folder's objects are and any attribute information that goes along with each object. Although many users wait until the folder is completely populated, you can start to work with the objects as soon as they appear in the folder. As more and more objects are added to a folder, the time to populate the folder with the objects will increase. It is recommended that you keep the number of objects in any folder to 40 or below.

Another small factor which affects folder population is related to the icons associated with program references. If the icon is part of the extended attribute information rather than part of the executable (.EXE) the icon will appear faster. This is due to the additional query time associated with searching the executable rather than searching an extended attribute list.

13.3.2.2 Desktop Considerations

Creating a folder with a shadow of commonly referenced desktop objects may increase your productivity. When there are multiple windows active on the desktop (applications or folders), it becomes increasingly difficult to navigate the Workplace Shell. By creating a folder with shadows of commonly referenced objects, it is easier to access the objects. You can keep the folder minimized and resurface it very quickly by using the Window List.

An alternative to creating a folder with shadows in it is to modify the desktop settings so that you can actually have multiple views of the desktop open at once. By changing the "Object open behavior" from "Display existing window" to "Create new window", you can open multiple desktops. This may come in handy if the desktop is becoming cluttered and you need to access a printer object. With the above settings in place, you could bring up the desktop pop-up menu and select "Open" and your choice of Icon view, Tree view or Details view. This will open a folder with the view of your choice and give you quick access to your objects.

Note: Selecting Arrange in this "new" window will also cause the "regular" desktop to arrange its objects. However, if the shadow technique is used, an Arrange will only affect the contents of the new folder with the shadow objects in it.

13.3.2.3 Menu Options and Considerations

One of the options on the pop-up menu for an object is "Create another". If you want to create a new folder, it is usually faster to select the Create another menu option rather than opening the Templates folder, finding the Folder Template, and dragging off a copy. Some of the other menu options which may aid in productivity are related to adding menu options to the pop-up menu.

It is possible to add menu options to the pop-up menus of the Workplace Shell. This may be useful to provide quick access to your favorite application. Use the following steps if you want to add an application to the desktop pop-up menu.

1. Open the Settings notebook for the Desktop.

2. Select the Menu tab.
3. With the "Primary pop-up menu" highlighted in the "Available menus:" list box, press the "Create another..." button next to the "Actions on menu:" list box (lower part of settings page).
4. Enter a name for your application as you want it to appear on the pop-up menu in the "Menu item name:" entry field.
5. Enter the physical path and name of the program in the "Name:" field of the "Program" dialog area (optionally, you could use the "Find program..." button to do a search for the program if the path is not known).
6. Select "OK" and verify that your entry appears in the "Actions on menu:" list box.
7. Close the Settings notebook.
8. Bring up the pop-up menu for the desktop and select your new menu option to load your application.

13.3.2.4 Startup Folder Considerations

Another way to increase your productivity is through the use of the Startup folder. Applications, folders, and objects which are desired to be started automatically should be placed in the Startup folder, which is located in the OS/2 System folder. This is very similar to activities in your own office. The most recent projects worked on are put in the top shelf of the file drawer. When you arrive in the morning the first things you pull out are the top items in the file drawer.

The objects in the Startup folder will start in a random order. If it is necessary to have certain applications start in order, there are other ways to achieve this. One way is to order the applications in a STARTUP.CMD file. Another way is to include RUN or CALL statements in the CONFIG.SYS file.

13.3.2.5 Object Associations

One easy way to customize the Workplace Shell is to make use of "Associations". This refers to linking applications and data objects by particular attributes, such as types or extensions. An example of this is to be able to double click a data object with a .XYZ file extension and have it automatically start and load the file into the PM Chart application. To accomplish this type of scenario, follow these steps:

1. Bring up the Settings notebook for the application for which you want to associate a data object (PM Chart for this example).
2. Select the "Association" tab of the Settings notebook.
3. Type "*.XYZ" in the "New name:" field and select "Add> >".
4. Your file extension should show in the "Current names" list box.
5. Close the Settings notebook.
6. Double click on an object with the .XYZ extension and watch the PM Chart application load.

13.3.2.6 General Considerations

If you are the adventurous type and want to explore the "settings" of the various objects and of the system, be sure that you know what you are doing. It is very easy to start making changes and not remember how to recover in the event of a failure. Changing statements in the CONFIG.SYS file should be left to an administrator or technical person. Changing some of the settings for an application can be performed with less expertise. Some of the items you may want to experiment with are:

- Full Screen versus Windowed Applications
- Display Existing Window versus Create New Window
- Creating Customized Templates
- Creating Shadows

The first two items in the list can be accessed by the Settings notebook for a particular application. Select the "Session" tab and the "Window" tab, respectively. Using and modifying templates can be done through the Templates folder on the default Workplace Shell desktop. Before modifying the default templates create copies and then modify the copies. This will insure that any errors can be recovered from by starting back at the beginning. A shadow of an object can be created by holding the Ctrl+Shift keys simultaneously while dragging an object to a new location. Further information on these topics can be found in the Master Help Index and the README file.

Since most users will have numerous objects open at any given time, it is helpful to use the minimize button and the Window List to ease navigation of the Workplace Shell and active applications and folders. Keeping applications and folders minimized makes finding desktop objects easier and also reduces the amount of processing OS/2 has to do when applications are open on the desktop.

13.4 System Configuration Considerations

The goal of this section is to present information to help the reader decide what factors are relevant in choosing a particular hardware platform. As stated in the announcements and on the packaging for OS/2 (any 2.x version), the minimum requirements are a 386SX or higher processor, and a minimum of 4MB memory. Although this is true, the following information shows that even *minimal* increases in processor or memory can *significantly* increase performance.

As in the previous sections which discussed performance, this is a very environment-relative discussion. The figures presented should only be used for information purposes. It is best to test several platforms for your particular operating environment.

A couple of notes about performance are warranted before discussion of the tests. Tests are very helpful in terms of providing input into formulas (structured or logical) for determining optimal computing environments. However, part of the difficulty of determining performance is that it is extremely difficult to isolate individual components of an operating environment. How can a particular gain or decline in performance be *accurately* assigned to a particular subsystem or component? This section cannot answer all claims about every subsystem, but it is helpful in terms of filling in some of the pieces and helping to provide a more complete picture.

13.4.1 Effects of Processor on Performance

With the rapid decline in the price of processor platforms and related subsystems, it is of interest to highlight performance of different processors in an OS/2 operating environment. What should one expect when upgrading from a 386SX to a 486SLC2, or 486DX to 486DX2 processor complex? This discussion about processors and processor complexes is intended to be an overview. Not all processors or variations are represented by these tests.

Table 25 illustrates the Intel and IBM microprocessor platforms and their basic characteristics.

Table 25. Intel and IBM Microprocessor Specifications

Micro-processor	Data Path	Processor	Address Path	Math Co-processor	Internal Memory Cache	Speeds (MHz)
80386 SX	16-bit	32-bit	24-bit	none	none	16,20
80386 SL	16-bit	32-bit	24-bit	none	none	25
80386 SLC	16-bit[1]	32-bit	24-bit	none	16KB	16,20,25
80386 DX	32-bit	32-bit	32-bit	none	none	16,20,25
80486 SX	32-bit	32-bit	32-bit	none	8KB	20,25,33
80486 SLC	16-bit[1]	32-bit	24-bit	none	16KB	25
80486 SLC2	16-bit[1]	32-bit	24-bit	none	16KB	40/20, 50/25[2]
80486 DX	32-bit	32-bit	32-bit	yes	8KB	25,33,50
80486 DX2	32-bit	32-bit	32-bit	yes	8KB	50/25, 66/33[2]

Note:

[1] Although SLC and SLC2 microprocessors have a 16-bit data bus, the memory cache is integrated on the processor chip with transfers from memory cache to CPU in 32-bits.

[2] The SLC2 microprocessor runs twice as fast internally (40MHz) as externally (20MHz).

The DX2 microprocessors have speeds of 50/25 MHz and 66/33 MHz.

Some of the factors within the processor area which may have an impact on performance are the processor speed, processor type, and data path width. The data path on SX processors is 16-bit and on DX processors it is 32-bit. In a 32-bit operating system, a 16-bit data path introduces a constraint for data transfer. The 32-bit memory model uses double word length data moves which require a 16-bit processor to break the double word into two words. The data is then pushed through the data path and concatenated at its destination.

Processor speeds also have an impact on performance. It is obvious that a 33 MHz processor can perform the same amount of work in less time than a 16 MHz processor. However, it is important to clarify some of the other processor

speeds. The 66/33 MHz processors (DX2) have two clock speeds. Instructions internal to the CPU (push, pop, and move types) are performed at speeds of 66 MHz while external instructions (fetch types) are performed at 33 MHz. It would be expected that a 66/33 MHz CPU would have higher performance than a true 33 MHz CPU. However, since more instructions are of the *external* type, the 66/33 MHz CPU will not perform fully twice as fast as a 33 MHz processor. Most tests have shown that the performance increase can be as high as 70%. Due to the external/internal instruction ratios, the 66/33 MHz processors will generally not perform as fast as a true 50 MHz processor.

13.4.2 Effects of Memory on Performance

OS/2 is an operating system with the concept of virtual memory at its core. When there is no physical memory available for processing, OS/2 will utilize space on hard disks to emulate memory. There is a performance cost associated with paging to and from hard disks. It is therefore very valuable to maximize the amount of memory on your system. Even incremental increases in memory can have dramatic impacts on performance.

13.4.3 Effects of DASD on Performance

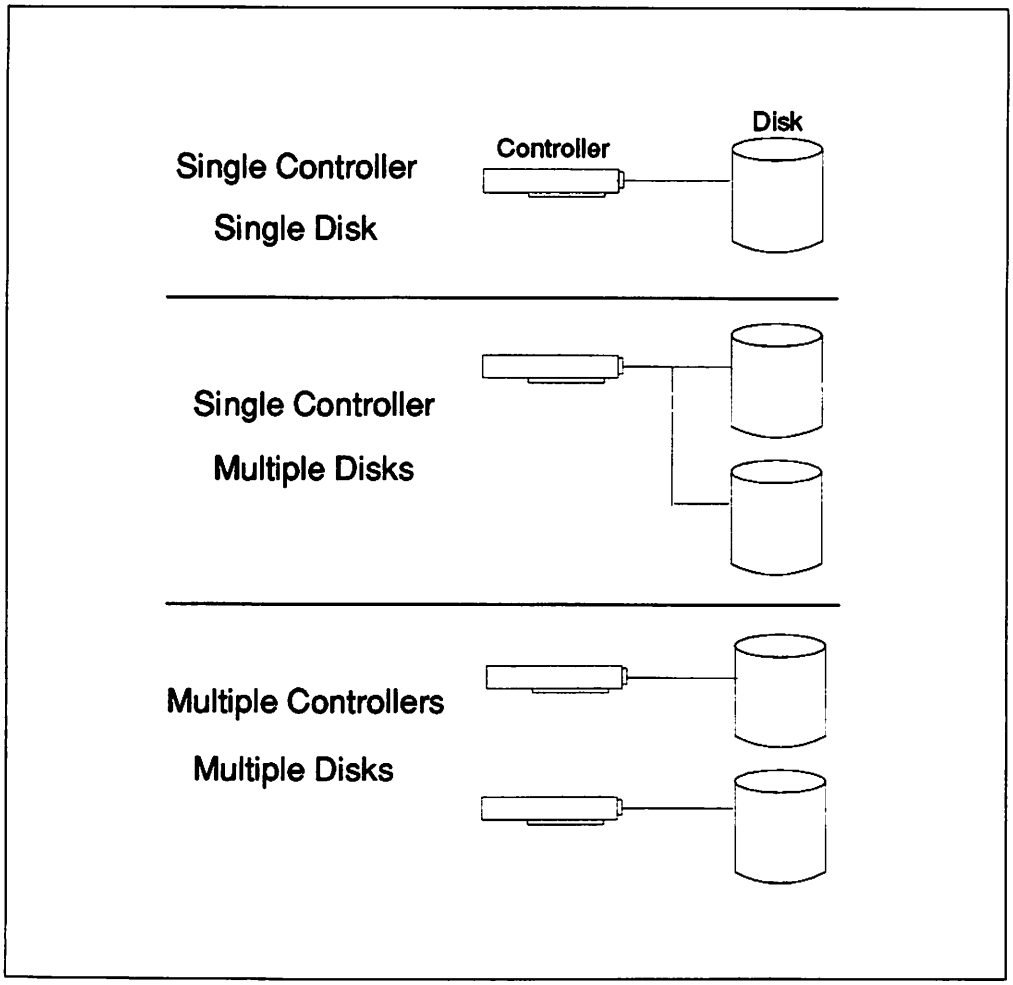


Figure 135. Overview of Disk and Controller Combinations.

It is not surprising that DASD plays a role in the overall performance since OS/2 is a paging-oriented operating system. There are differences in the DASD

architectures which may have impacts on performance in your environment. One of the factors related to DASD is the number of hard disk controllers in your system. Most computers come with one controller and one hard drive. The requirements of loading applications and data, spooling print output to devices, and OS/2's memory paging place quite a burden on a single disk controller.

Figure 135 on page 195 illustrates some of the combinations of controllers and hard disks. In this example, controller refers to the adapter card which supports hard file connections, such as IBM's SCSI Adapter with Cache. Controllers can also be integrated on the planar board of the computer. Having only one controller can introduce a constraint on the system in an I/O intensive environment.

Placing a second disk controller can ease the I/O burden of your system and have impacts on performance. Additionally, busmaster cards are available for some computer systems that will also improve performance. A busmaster card contains a processor on the card which helps to offload the main CPU of I/O related processing. Additionally, SCSI devices support I/O overlap which means that while one controller is busy processing, the other controller can be handling other interrupts for the file system.

13.5 Enhancements and Performance Impacts

With preloaded OS/2 2.00.1, the Service Pak XR06055 for OS/2 2.0, and OS/2 2.1, there have been numerous functional enhancements to OS/2. Additionally, the developers of OS/2 have done extensive tuning of the existing base operating system to yield performance enhancements. This chapter will recap the enhancements to OS/2 Version 2 and discuss their respective performance impacts.

A few notes about performance in general are warranted before discussing performance impacts of particular changes. It is important not to lose sight of the overall system performance, sometimes termed throughput, rather than focusing on individual performance savings or costs. A general example of this would be using a 32-bit SVGA adapter that may have an increased working set requirement but has better performance throughput due to its 32-bit capability. How does one weigh the actual performance improvement or decline (throughput vs. working set increase)?

The releases of OS/2 (and their respective changes) to be discussed are:

- Preloaded OS/2 2.00.1
- OS/2 2.0 with Service Pak XR06055 applied
- OS/2 2.1

Some of the changes which came about in these releases are:

- 32-bit graphics engine
- WIN-OS/2 3.1
- File system changes
- Loader and scheduler changes
- Paging management changes

This section summarizes these changes and discusses how performance may be impacted.

13.5.1 32-Bit Graphics Engine

The primary change in preloaded OS/2 2.00.1 is the introduction of the 32-bit graphics engine. This change is very profound in terms of the "capacity" the operating system now has for performance improvements. The word capacity is used deliberately, since actual performance gains depend primarily on your operating environment such as your application types or graphics adapter type.

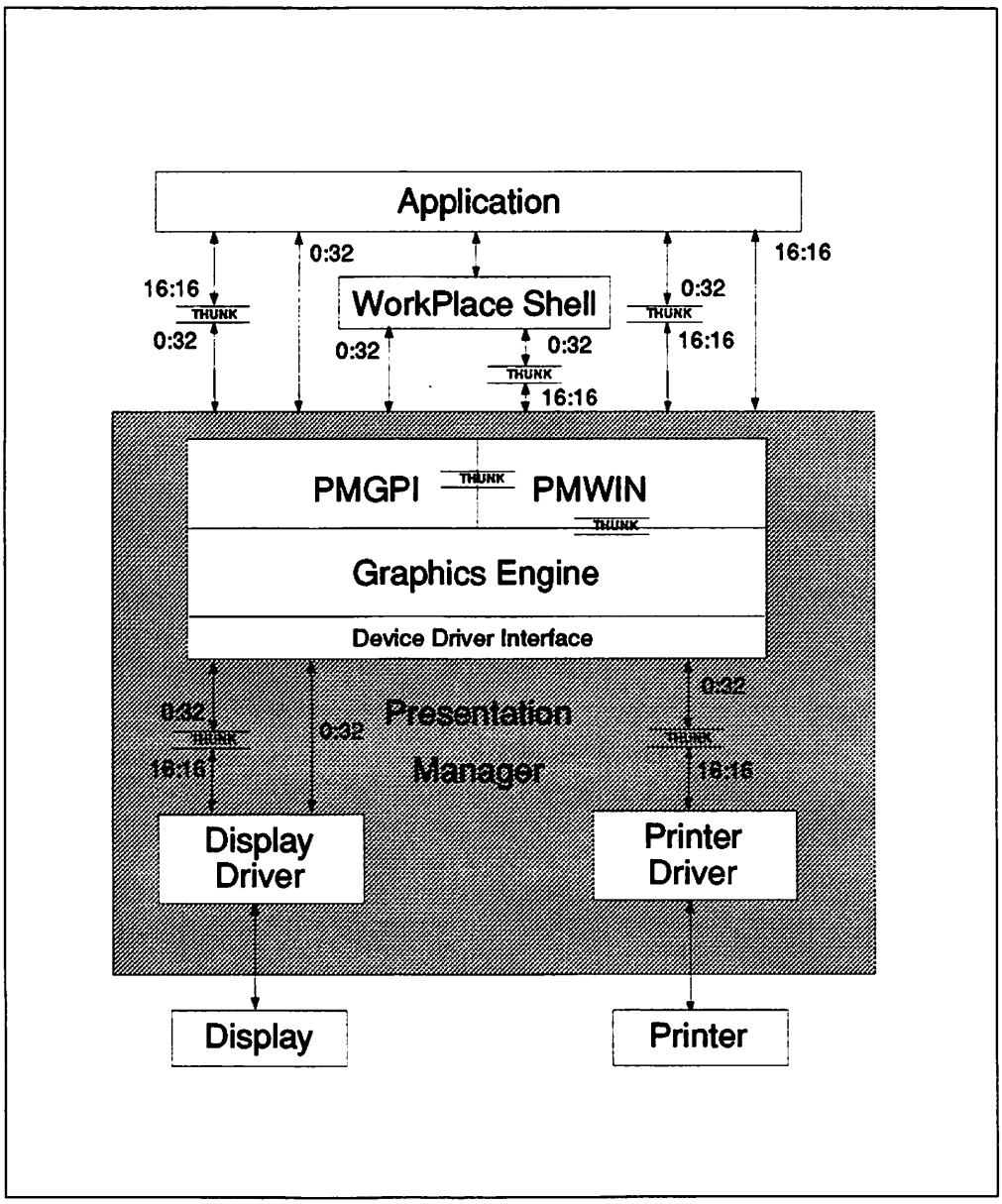


Figure 136. Presentation Manager with 16-Bit and 32-Bit Application Flows

As illustrated in Figure 136, there are numerous factors that may affect throughput. The type of action being performed, the type of computer, the type of graphics adapter or printer driver are the primary influences on throughput. For a detailed discussion of the architectural structure, please refer to Chapter 8, "Presentation Manager Enhancements" on page 105, and also to

OS/2 Version 2.0 - Volume 3: Presentation Manager and Workplace Shell, GG24-3732.

Tracing through an application flow helps to highlight different performance scenarios and put impacts on application usage and application development into perspective.

An example of a best-case scenario would be a 32-bit OS/2 application writing to a 32-bit device driver through the 32-bit graphics engine. For illustration purposes, our sample application will draw an outline of the world containing 8 million points.

In essence, the application makes a call to the 32-bit graphics programming interface (PMGPI) which is processed via the 32-bit graphics engine. The graphics engine writes to the presentation display driver and on to a 32-bit display adapter. When the final translation of the memory address to pixels is made for the display, the original address of the application's memory object is exactly the same as the display driver.

No translation or offsets are necessary since this is a 32-bit environment and the 0:32 memory address in the application maps exactly to the 0:32 memory address passed to the display. When the application draws the map of the world outline with 8 million points, the whole array of points can be sent as one memory address space.

On the contrary, a worst-case scenario would be similar to the flow above, however with a 16-bit application and a 16-bit device driver. The application would encounter two "thunk" layers on the way to the device driver. Thunking refers to the translation of segmented memory structures with 16-bit addresses and 16-bit offsets (noted as 16:16) to the 32-bit flat memory structure (noted as 0:32). The first address translation is at the application and GPI level and the second is between the graphics engine and the device driver.

In the case of printer drivers (most of which are still 16-bit) there may be an additional thunk layer due to the synchronous communication between the device driver and the graphics engine. The example of the map above would require the 8 million points to be transferred in 64KB segments due to the 16-bit addressing. In this case, the application enters a loop until all the points have been processed by these segments. An application which utilizes the flat memory model and 32-bit addressing will usually provide better performance than an application flow with thunking of 16-bit addresses.

For additional information on thunking and the thunk layer, please refer to *OS/2 Version 2.0 - Volume 1: Control Program, GG24-3730.*

An additional change has been made to the device driver interface within the graphics subsystem. The algorithm which determines the drawing of shapes to the device driver has been modified. This can be thought of as a "cache" of sorts, with common shapes or objects resident, and their clipping at the device can be accomplished in shorter time. For instance, what may have taken 45 pages of memory in OS/2 2.0 might only take five pages for the same shapes to be drawn in OS/2 2.1. This impacts performance whether 16-bit or 32-bit device drivers are in use. Please see 8.2, "32-bit Graphics Engine (PMGRE)" on page 106 for additional information on the 32-bit graphics engine.

13.5.2 High Performance File System Changes

The basic structure of the High Performance File System (HPFS) remains unchanged, except for a few modifications. First, the Read Ahead feature has changed from Strategy 1 to Strategy 2, implying support for *scatter/gather*. This is the capability to read data from physically discontiguous pages in a single I/O operation. This is a hardware feature supported on DMA devices (such as IBM SCSI adapters) and is now exploitable on HPFS volumes. The enhanced read ahead should increase performance for HPFS formatted drives capable of supporting DMA scatter/gather.

Additionally, HPFS has an enhanced file caching algorithm. HPFS looks at the I/O read operation and if there is enough supporting cache defined to the file system it will try to pull the whole file into the cache. This should yield performance gains in file-sharing environments such as local area networks. The table below outlines the new algorithm and the supporting file sizes to be cached.

Table 26. Cache Sizes and Supporting File Read Limits

Cache Size (specified in CONFIG.SYS)	Largest File Size Fully Readable into Cache
> 1MB	128KB
512KB	64KB
256KB	32KB
128KB	16KB

By specifying a cache greater than or equal to 1MB (see 13.2.9, "Standard CONFIG.SYS Tuning Parameters" on page 183 for this specification), files of 128KB and smaller will automatically be cached in their entirety. This can dramatically increase application performance. Additionally, the table suggests that gains can be made in *memory constrained* environments. With as little as 128KB specified for the cache, files of 16KB or less will be brought into the cache in their entirety.

13.5.3 Paging Management Changes

With OS/2 2.1, a new COMMIT parameter has been introduced for the SWAPPATH statement. If COMMIT is used, then backing store will be reserved in the SWAPPER.DAT file for allocated but uncommitted memory, at the time of the memory allocation. This parameter may be needed in order for some 16-bit OS/2 applications to run; it is not the default and will usually cause a larger than normal size for the SWAPPER.DAT file.

13.5.3.1 Review of SWAPPER.DAT Behavior

In order to understand this change, it is useful to first review in detail the default behavior of SWAPPER.DAT. This is the standard behavior for OS/2 2.0, and also the behavior for OS/2 2.1 if the COMMIT parameter is not specified.

The paging function of OS/2 includes a "MINFREE" parameter on the SWAPPATH statement in the CONFIG.SYS file. The MINFREE specifies a desired amount of disk space on a partition which, if written into by the SWAPPER.DAT file, would trigger a warning dialog box to the effect of "The partition with the SWAPPER.DAT file is full. Please close applications or free additional space on the partition."

Warning!

When this warning message appears, you should close some applications or free some disk space by deleting unwanted files before continuing. If you ignore the warning message and continue working then further messages will be generated, and eventually the system will stop (when there is no more disk space left for the SWAPPER.DAT file to expand into). You will need to power-off and power-on the system to resume operation, and you are likely to have lost or corrupted some of your data.

Consideration should be also be given to avoiding this from happening, by placement of the SWAPPER.DAT in a partition with plenty of available space (or in its own partition) such that intensive paging conditions can be satisfied. This can now be done at installation time.

The MEMMAN statement in Figure 137 specifies that swapping is enabled (SWAP) and DLLs may use protected memory (PROTECT). The SWAPPATH statement specifies the location of the SWAPPER.DAT file (D:\OS2\SYSTEM), the MINFREE value (4096 (4MB)) and the initial SWAPPER.DAT allocation size (2048 (2MB)).

```
MEMMAN=SWAP,PROTECT
SWAPPATH=D:\OS2\SYSTEM 4096 2048
```

Figure 137. MEMMAN and SWAPPATH Statements

If the SWAPPER.DAT file tries to grow into the area of the partition specified by the MINFREE amount, then the user would see warning messages that the MINFREE area has been entered. Figure 138 on page 201 illustrates the remaining free space on a partition, and the progression of dialog boxes the user would see.

As a reminder, the initial allocation size parameter indicates how large the allocation for the SWAPPER.DAT file will be at startup. This may provide slight performance improvements when it is known what the paging environment is like. For instance, if the system has 6MB of memory, and the working set normally operates at 8MB, the initial allocation size of the SWAPPER.DAT file should be at least 2MB. The preallocation of the SWAPPER.DAT file will reduce disk query and allocation time when the SWAPPER.DAT file eventually has to grow. Preallocating a large swap space may also aid in reducing disk fragmentation.

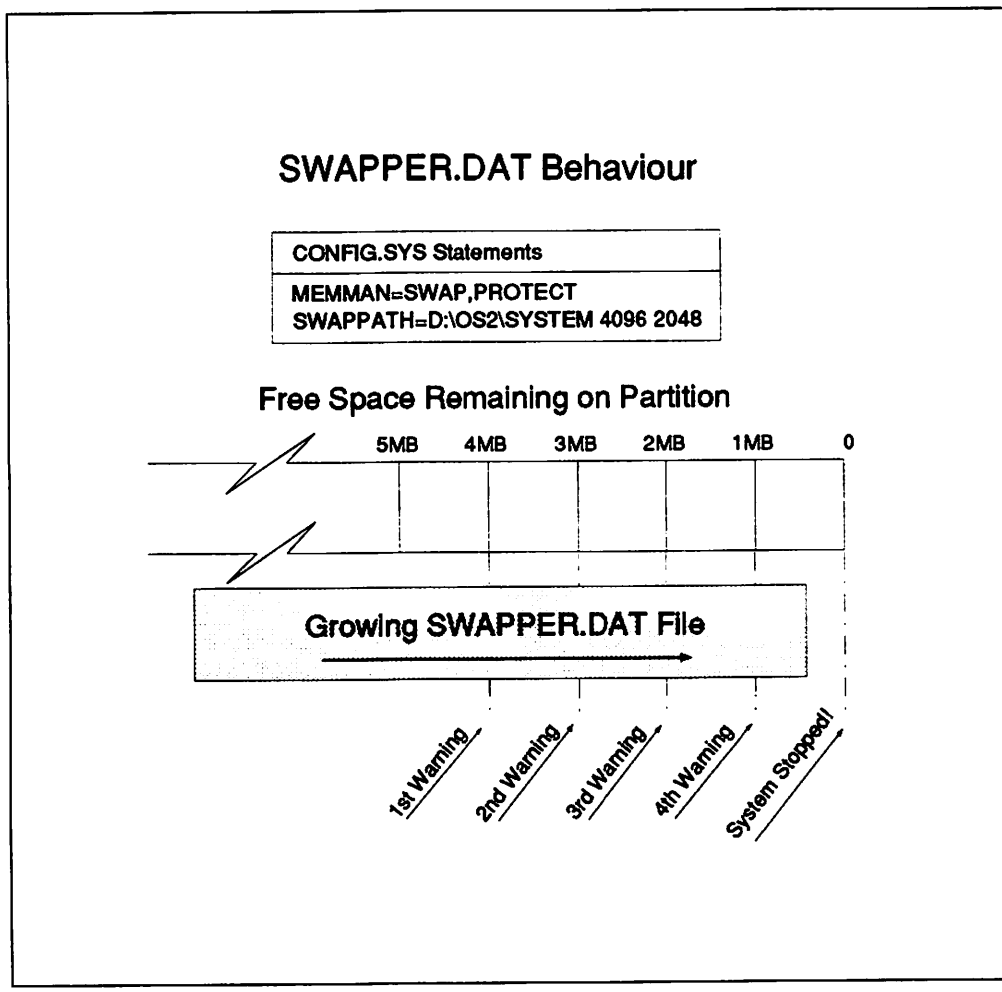


Figure 138. SWAPPER.DAT Growth If Warning Messages Ignored

Additionally, there have been numerous ideas as to where to place the SWAPPER.DAT file. To determine the location of your SWAPPER.DAT file, consider the following factors:

- Number of disk controllers
- Activity on controllers

Notice that the "controller" is the key to the equation. Controller refers to an adapter card or a device on the planar board of the system. There is sometimes one controller for each physical hard drive in the system that manages the I/O operations to and from the hard disk, but in more recent systems that use SCSI or ESDI, one controller can manage many hard disks.

If the environment is a heavy paging environment (such as heavy multitasking or constrained memory), the SWAPPER.DAT file will play an integral role in the system's performance and should be placed on a controller/disk combination not used by applications that are input/output intensive. The response time for writing to or retrieving from the disk on the least used controller will most often be less than competing with other I/O requests on a busy controller. This will provide the best response when having to "swap in" pages which were previously "swapped out" to disk from memory.

13.5.3.2 Use of the COMMIT Parameter

OS/2 2.0 introduced the idea of being able to allocate memory without committing it. This works fine for 32-bit OS/2 applications; backing store is allocated in the SWAPPER.DAT when the application commits the memory. If at this stage there is insufficient disk space for the SWAPPER.DAT file to grow, then the application will receive an error return code at commit time.

However, some 16-bit OS/2 applications have experienced problems because there was no concept of sparse memory allocation in OS/2 1.x. Sparse memory allocation is implemented by OS/2 Version 2 transparently to the 16-bit allocation; memory is only committed when its page is accessed. This can cause problems, since the 16-bit application will have assumed that all the memory it has asked for is committed at allocation stage. When the application tries to access the memory, if the system is unable to grow the SWAPPER.DAT file any more (due to disk space limitations), then the system has no way of telling the application that it cannot now provide the memory, and the application will try and access memory that is not there and it or the system will fail.

The COMMIT parameter forces backing store for memory to be allocated in the SWAPPER.DAT file at the time of allocation. If there is not enough room on the disk for the SWAPPER.DAT file to grow, then the application will get an error return code from the system at this stage, and will know that its request has been refused. Thus an application will fail to load because of insufficient memory, rather than failing mysteriously whilst running.

```
MEMMAN=SWAP,PROTECT,COMMIT
SWAPPATH=D:\OS2\SYSTEM 4096 2048
```

Figure 139. MEMMAN and SWAPPATH Statements With COMMIT Parameter

Although this works well for 16-bit OS/2 applications, this is a system-wide parameter and will affect 32-bit applications as well, since any sparse memory they request will have backing store in the SWAPPER.DAT file reserved at the time of allocation rather than when they commit the memory.

Since this backing store may lead to a very large SWAPPER.DAT file, we recommend that the COMMIT parameter is not used unless it is needed for a specific 16-bit OS/2 application to work, and then it is used with care. The default OS/2 configuration does not use the COMMIT parameter.

13.5.4 WIN-OS/2 3.1

OS/2 2.1 includes support for Windows 3.1 applications through WIN-OS/2 3.1. Performance has been improved due to the intrinsic performance increases in Windows 3.1, and also through changes to the OS/2 and WIN-OS/2 code.

Components of OS/2 that provide improved WIN-OS/2 support include the display drivers for the seamless graphics, which have been optimized for quicker paint and draw functions.

13.5.4.1 WIN-OS/2 3.1 DDE and Clipboard Performance

A function of WIN-OS/2 which has been enhanced is the data exchange facility. This changed in two areas: Virtual Device Drivers (VDDs) and Virtual Rendering. The WIN-OS/2 Dynamic Data Exchange (DDE) under OS/2 2.0 uses named pipes to communicate between client and server. WIN-OS/2 3.1 uses VDDs to accomplish DDE with reduced overhead and better integrity than was possible with the named pipes. Physical device drivers (PDDs) provide access to devices within OS/2 and only one process can "own" a physical device. Virtual device drivers allow many processes to share the PDD and device by providing a management layer on top of the PDD.

The clipboard is a storehouse for data exchange. Under OS/2 2.0, data copied into the clipboard was converted to all possible data formats which the application was able to render, since the data formats supported by the client or destination application were not yet known. This multiple-format conversion caused a lot of overhead on the system. The clipboard with WIN-OS/2 3.1 uses a *virtual rendering* of the data. This means that the data is stored in the clipboard once and converted to the client format only when the client requests it. Only one conversion is necessary, and this should save time and processing for data exchange and other operations.

13.5.4.2 WIN-OS/2 3.1 Memory Management Considerations

Many sections in this chapter discuss memory management from an OS/2 perspective and the importance of memory management to system performance. WIN-OS/2 3.1 memory management can also have significant impacts on performance. Memory allocation in WIN-OS/2 3.1 sessions differs from OS/2 sessions and is dependent on the characteristics of the WIN-OS/2 3.1 kernel and the session settings for particular WIN-OS/2 3.1 sessions.

When a session is started in WIN-OS/2 (OS/2 2.0), the WIN-OS/2 kernel checks to see how much memory is available to it, as defined by the combination of DOS_RMSIZE and DPMI_MEMORY_LIMIT. However, the WIN-OS/2 kernel is intelligent and will only commit the memory which is needed, rather than all the memory allocated. Since the DPMI default for WIN-OS/2 3.1 is 64MB, this avoids large amounts of memory being committed which are potentially never used.

Another area of concern for WIN-OS/2 memory management is the type of windowed or seamless session in which your applications are to be run. An application can run in a "common" or in a "separate" seamless session, with common being the default type of session. Common seamless sessions share one WIN-OS/2 kernel, regardless of the number of applications being loaded. In contrast, separate sessions load a new copy of the kernel each time a session is started. This overhead can have a negative impact on performance if you run many separate sessions. However, the reason you may run applications in separate sessions is for the memory protection and system integrity that they provide. One badly behaving application will not crash all of the other Windows applications being run in separate sessions.

Figure 140 on page 204 illustrates the memory characteristics of running common and seamless sessions. The separate seamless example shows how each separate session allocates its own memory while the common seamless session example shows how the memory allocation is shared.

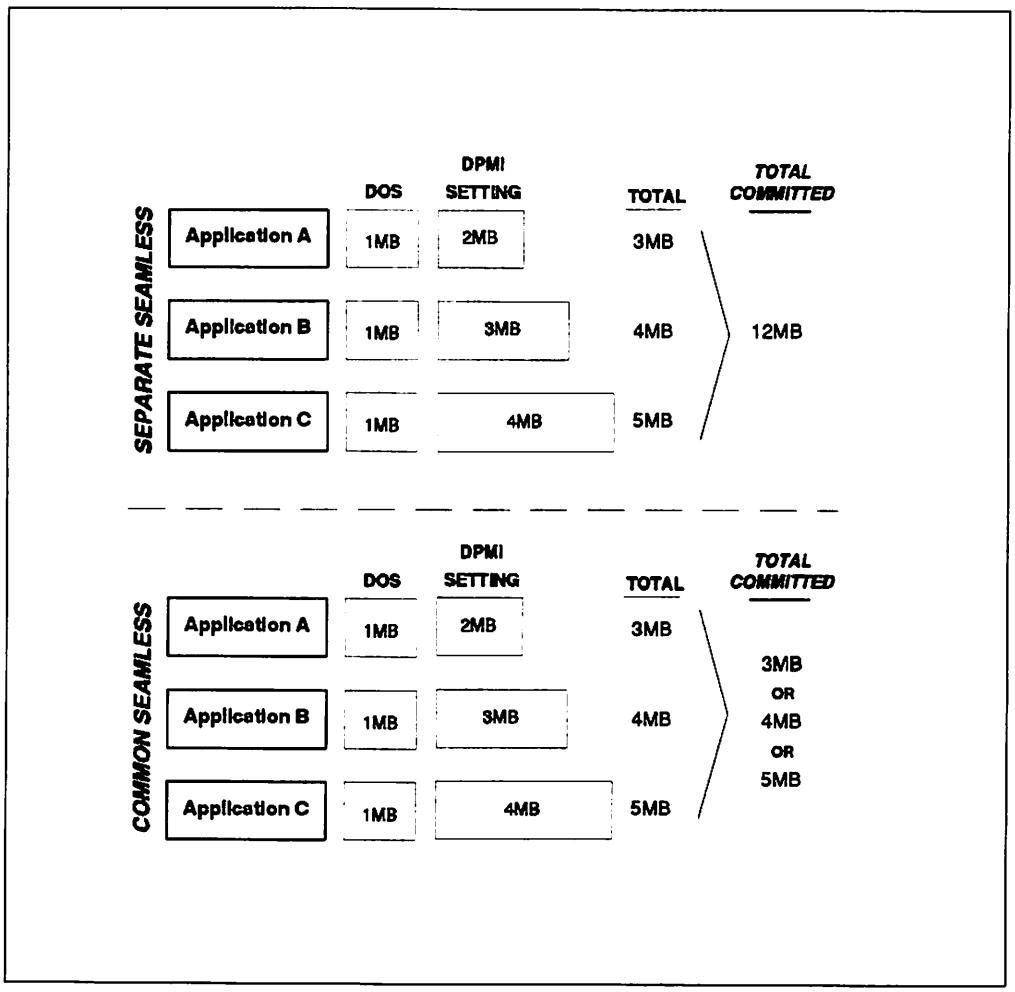


Figure 140. Separate and Common Seamless Session Memory Allocation

Note: A general assumption made for the examples in Figure 140 and Figure 141 on page 205 is that the applications can utilize DPMI memory.

When Applications A, B and C are each run in separate sessions, the total memory allocation would be roughly 12MB. In the separate seamless session example, it does not matter which application loads first because all applications will go through the same process: loading DOS, loading WIN-OS/2 kernel and loading the application. "Total Committed" shows the combined amount of memory specified on each sessions' settings.

On the other hand, when Applications A, B, and C are run in common seamless sessions, the amount of memory allocation will depend on the settings of the *first* application to load. The settings for Application A will set the WIN-OS/2 environment if it is loaded first. In this case the total memory allocation will be roughly 3MB (2MB for DPMI and 1MB for DOS; the extra overhead is not taken into account in these examples). With these settings, Applications B and C will *share* this environment with Application A when they are loaded. The DPMI settings for Applications B and C are ignored. This is illustrated in Figure 141 on page 205.

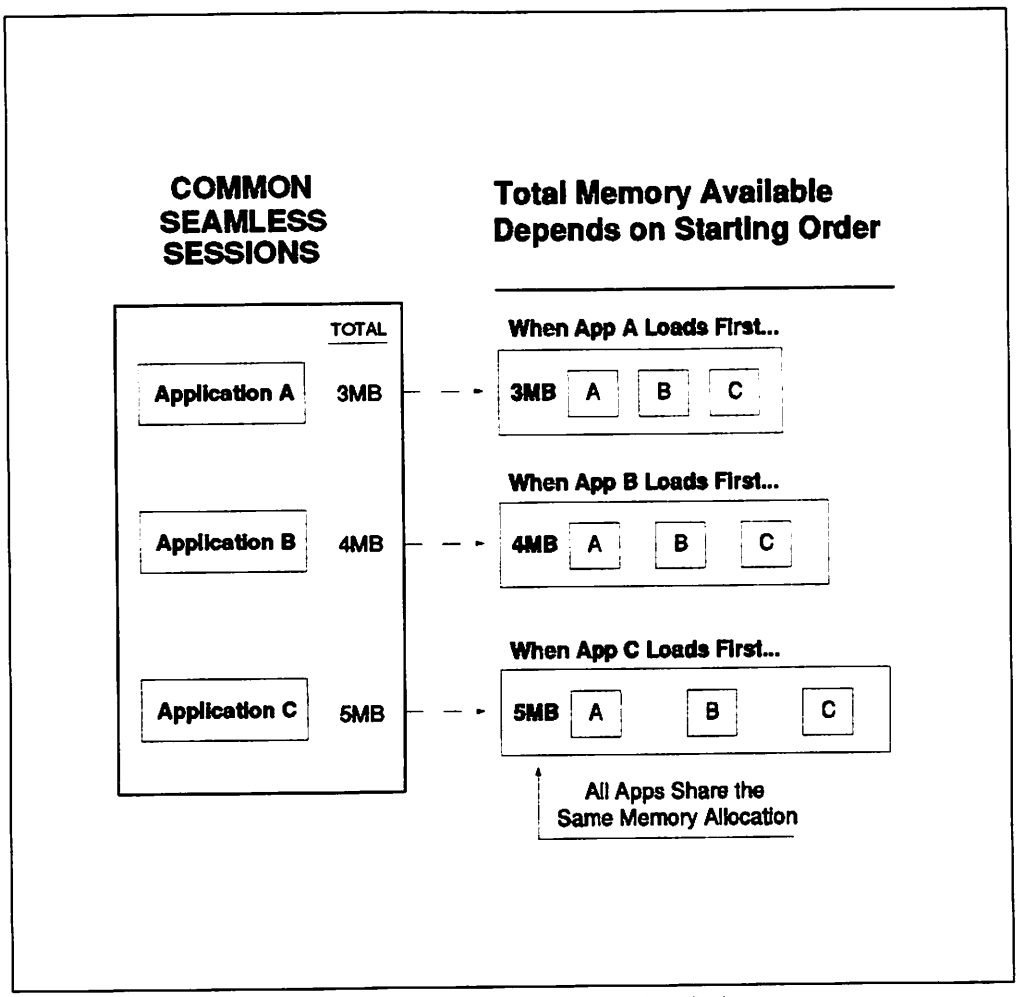


Figure 141. Common Seamless Session Memory Allocation Order

There may not be enough memory available for all three to run properly if these are large applications. It is important to understand the requirements of each application and the order in which they start. The benefit of running applications in common seamless sessions is the reduced overhead of having numerous WIN-OS/2 kernels loaded. The potential problem with running in common seamless sessions is the lack of integrity and crash protection due to the shared memory allocation. If one application has a general protection fault and crashes, the WIN-OS/2 session may be stopped without regard to the other two running applications.

Selecting common or shared seamless mode is done through the settings notebook for the application. Check the radio button for "Separate session" on the "Session" page of the notebook. Note that "Separate session" is only an option for the seamless ("WIN-OS/2 window") sessions.

13.6 Internal OS/2 Performance Tuning

Performance Disclaimer

The performance charts in this section have been compiled from performance tests run by the Personal Systems Programming development laboratory at Boca Raton in Florida. These tests have been run in a controlled environment, and therefore the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

In addition, performance data varies between applications and between different hardware configurations. The performance results presented here are for some commonly used applications running on some typical hardware configurations.

Since the release of OS/2 2.0 in March, 1992, IBM has continued to work on its 32-bit operating system. Its efforts have been divided into three main areas: *performance tuning*, *functional enhancements* and *functional fixes*. These are *internal* to the operating system, implying that these are not areas the user or administrator can modify. For performance improvements, most of the work effort has focused on *page tuning* and memory management, which are discussed throughout this chapter. Additionally, IBM has enhanced many of the existing operating system functions to provide greater levels of internal efficiency and integration among subsystems such as heap management and file systems. Functional enhancements include improving and adding facilities to OS/2 2.1.

Some of the areas that have been improved are related to the internal structure of the OS/2 kernel. Some of the pages of the kernel have been reclassified from resident to pageable. This means that some kernel functions that are used infrequently can be paged to disk in order to free memory for other uses. Additionally, some of the pages that are required for starting the operating system are now pageable once OS/2 is started. These pages were previously part of resident memory without providing any functions after OS/2 was initialized.

Other areas that have been improved are page tuning and memory management. There now follows a discussion of page tuning, memory management, and heap management. These areas have received a lot of focus by the development teams at IBM and are important in understanding the commitment and resource IBM has invested in OS/2.

Note

The following sections relate to IBM's efforts to modify the internal structure of OS/2. These areas cannot be modified by an administrator or user. However, application developers may make use of some of these techniques to ensure the application code they develop is "efficient" in terms of memory usage and performance.

13.6.1 Working Set Memory

In an operating system with a linear (flat) memory model, as OS/2 2.0 and OS/2 2.1 implement for the Intel platform, the primary facility of data movement into and out of memory is through 4KB pages. In very simple terms, when OS/2 has to perform a task it demand-loads pages of code and data from disk into main memory. It then uses these *memory objects* for processing. In memory constrained situations, unused or infrequently accessed pages are discarded or swapped out to disk to make room for newly accessed pages to be loaded into memory.

Figure 142 illustrates a conceptual view of the working set memory. Given the amount of physical memory installed in the computer (RAM), a memory object called by the operating system is "located" in one of the following areas: Idle, Working Set, or Resident. **Idle** implies the memory object was used for processing at one point, but has not recently been used and can be discarded or swapped out to disk. **Working Set** is similar in terms of discardability or swappability, but the system is using or has recently used the memory objects for processing. Pages of memory objects can be discarded or paged out if the system's physical memory is overcommitted and additional memory objects are being called by the system. **Resident** refers to memory objects which are "fixed" in memory (such as parts of the OS/2 kernel or cache). These cannot be swapped out or discarded.

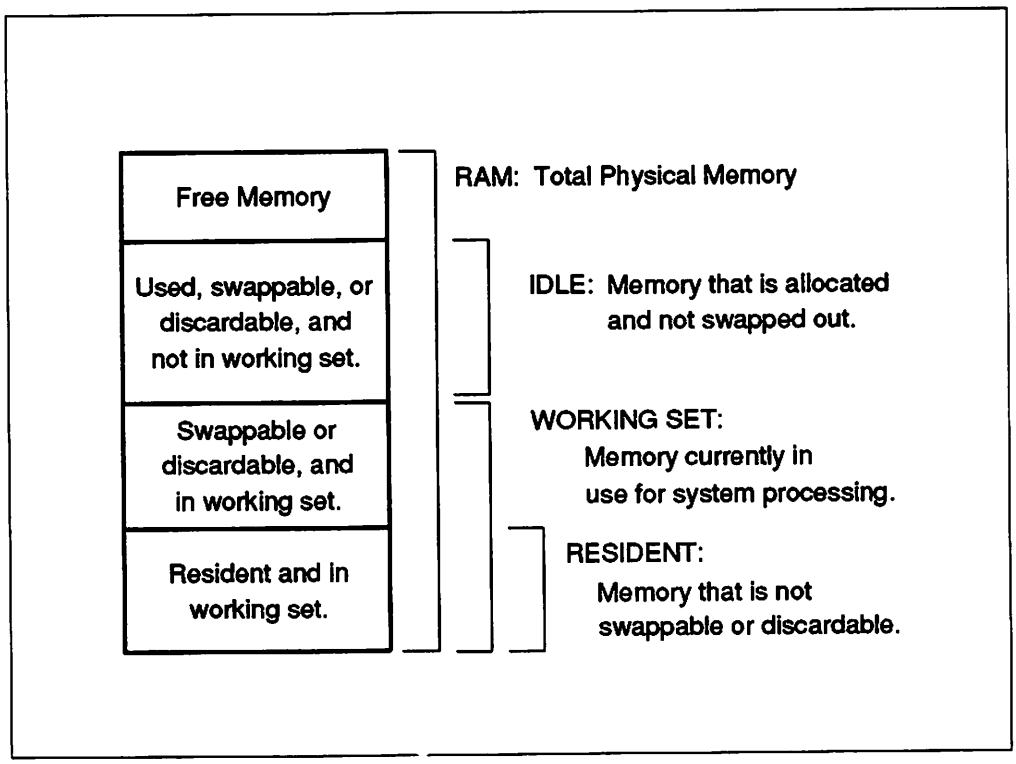


Figure 142. Conceptual View of Working Set Memory

Note

This is not a linear or physical view of OS/2's memory. What is grouped above for illustration may actually be spread over the entire memory structure. For further explanations of OS/2 and its memory structure and management, please refer to *OS/2 Version 2.0 - Volume 1: Control Program*, GG24-3730, and *OS/2 Version 2.0 - Volume 4: Application Development*, GG24-3774.

13.6.2 Page Tuning

Extensive tests are done to optimize OS/2 to determine what routines or functions are being called most frequently and how they are called. Page tuning then attempts to group those routines or functions together into the fewest number of pages. When a function call is made, the operating system is then able to accomplish the task with increased efficiency in terms of performance and memory usage. The same is true for applications if they are developed efficiently. For example, the Workplace Shell and the graphics engine are very critical in an object-oriented graphical user interface like OS/2. The end-user's perception of the system performance and responsiveness will improve if these components are page tuned properly.

Note

The terms paging and swapping are used interchangeably throughout this chapter and this section. Either term is used for describing the movement of one or more 4KB pages from main memory to disk and from disk into main memory.

The goal of page tuning is to reduce the *working set* in memory. In simple terms, the working set is the amount of memory being used for processing for any set time interval, including the amount of memory occupied by "resident" memory. System functions such as parts of the OS/2 kernel and parts of device driver memory objects would be "resident". Subtracting the working set from total physical memory yields the amount of available memory that other applications have to use. With many active processes, the working set increases up to the point where all available memory is "consumed." Any further requests to allocate and commit memory will cause pages of the working set to be swapped out to disk (SWAPPER.DAT file) or discarded. When it is time for the original process to continue, some of these memory pages have to be paged back into memory from disk.

This last scenario should be avoided if possible. The simple calculation of time to fetch pages from memory versus from disk illustrates why paging should be minimized. Memory is accessed at speeds on the order of nanoseconds, whereas accessing hard disks is on the order of milliseconds - *many orders of magnitude difference*! If you do not have to go to disk for data, the process is much faster.

The internal structure of OS/2 can be page tuned which leaves more physical memory available for applications to use. When an application can use only physical memory instead of using memory *and* disk, the application (or system throughput) will be faster. The extreme difference may not be perceptible if

paging only occurs infrequently. However, in most virtual memory environments paging happens continuously and has a negative impact on performance.

To highlight the effects of page tuning on reducing the working set, the following examples are presented. Please note the disclaimer concerning the data presented. As mentioned above, working set is measured over a time period. There are tools available to help understand your system's resource utilization, such as System Performance Monitor/2.

Between actual releases of OS/2 to customers, IBM creates numerous interim "builds", or development-level releases of OS/2. Extensive tests provide input for the page tuning which is done between these development-level releases. IBM uses function and routine traces at the hardware level to provide input to the tuning process. Each development-level release is page-tuned for performance and memory management and the testing begins again when the next development-level release is created. This iterative process of page-tuning development-level releases (or "builds") happens continuously up to the point of releasing a production-level build for customers.

Disclaimer!

These values should be used only to illustrate the value of page tuning for performance improvements and the resource IBM devotes to make OS/2 more efficient. The data was collected from development-level and production-level code and the actual values may change. As a reminder, these are working set sizes which can vary from scenario to scenario, and the data should be interpreted accordingly.

The data presented in this section is averaged from numerous sample scenarios. The applications involved simulated average work scenarios, such as spreadsheet, word processing, and Workplace Shell activity. The working set will be different for different scenarios, thus the need for the averaging of several scenarios.

There are two primary scenarios presented in the following figures. Figure 143 on page 210 is an "application intensive" scenario, with most activity focused on performing tasks and running macros within a few applications. Figure 144 on page 211 illustrates a "Workplace Shell intensive" scenario with most activity involved with manipulating folders and navigating the Workplace Shell.

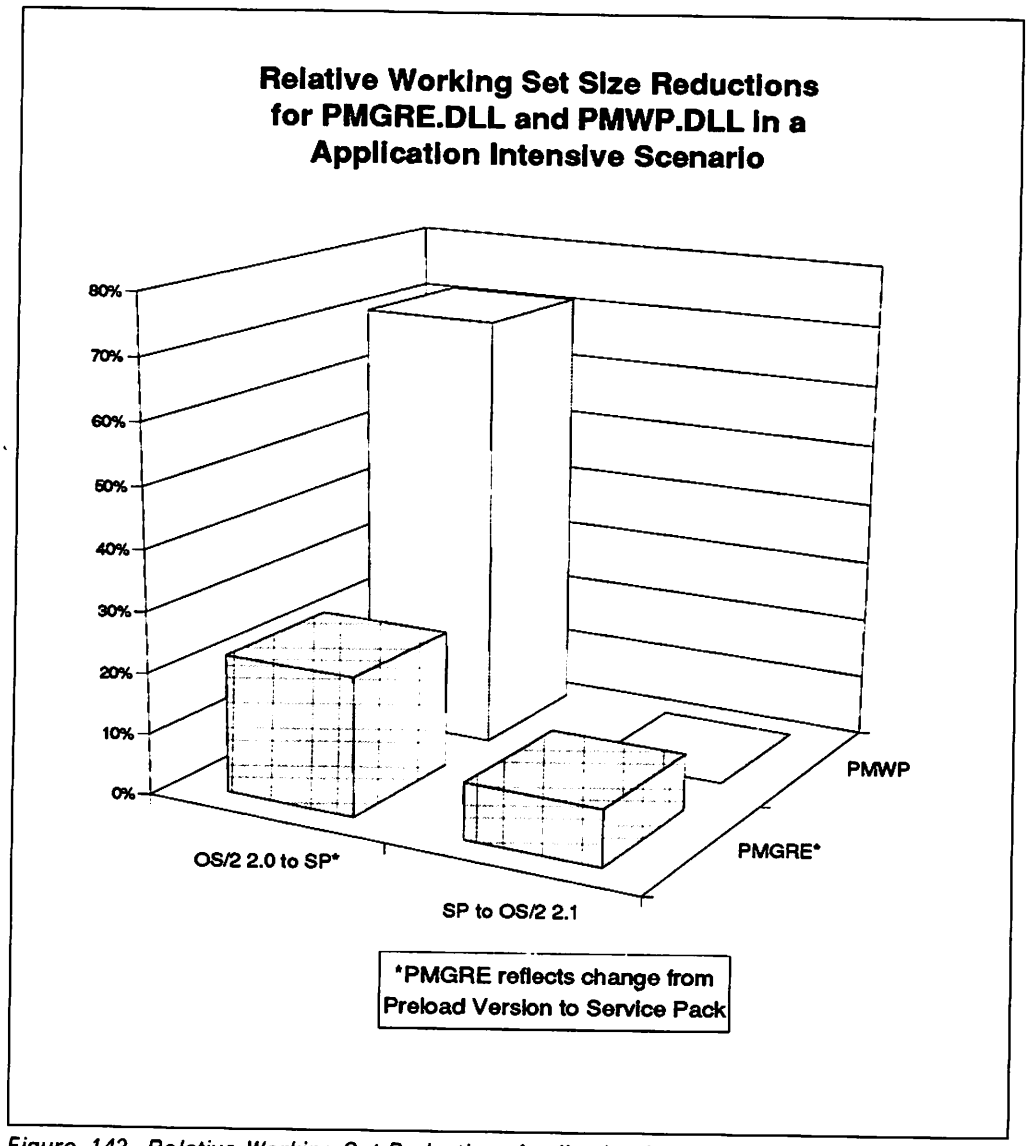


Figure 143. Relative Working Set Reduction: Application Intensive Scenario

Much tuning effort went into the graphics engine and the Workplace Shell. Figure 143 shows the relative reductions in working set through the efforts of page tuning for two of the DLLs. From Preloaded OS/2 2.00.1 to the release of the Service Pak XR06055 for OS/2 2.0, the PMWP.DLL and the PMGRE.DLL modules were reduced nearly 71% and 23% respectively, for application intensive scenarios. From the release of the Service Pak XR06055 to OS/2 2.1, the PMWP.DLL module remained the same and the PMGRE.DLL module was reduced an additional 9%. The large reduction in the working set size of the PMWP.DLL module in the Service Pak XR06055 is primarily from the page tuning of the Workplace Shell and how it interacts with the running applications. After the Service Pak XR06055, the tuning of the PMWP.DLL module yielded little reduction in this type of "application intensive" scenario.

Note

OS/2 2.0 implemented a 16-bit graphics engine. This was replaced with a 32-bit graphics engine beginning with preloaded OS/2 2.00.1 and on through OS/2 2.1. The data in these examples for the PMGRE.DLL module represent figures for only the 32-bit graphics engine.

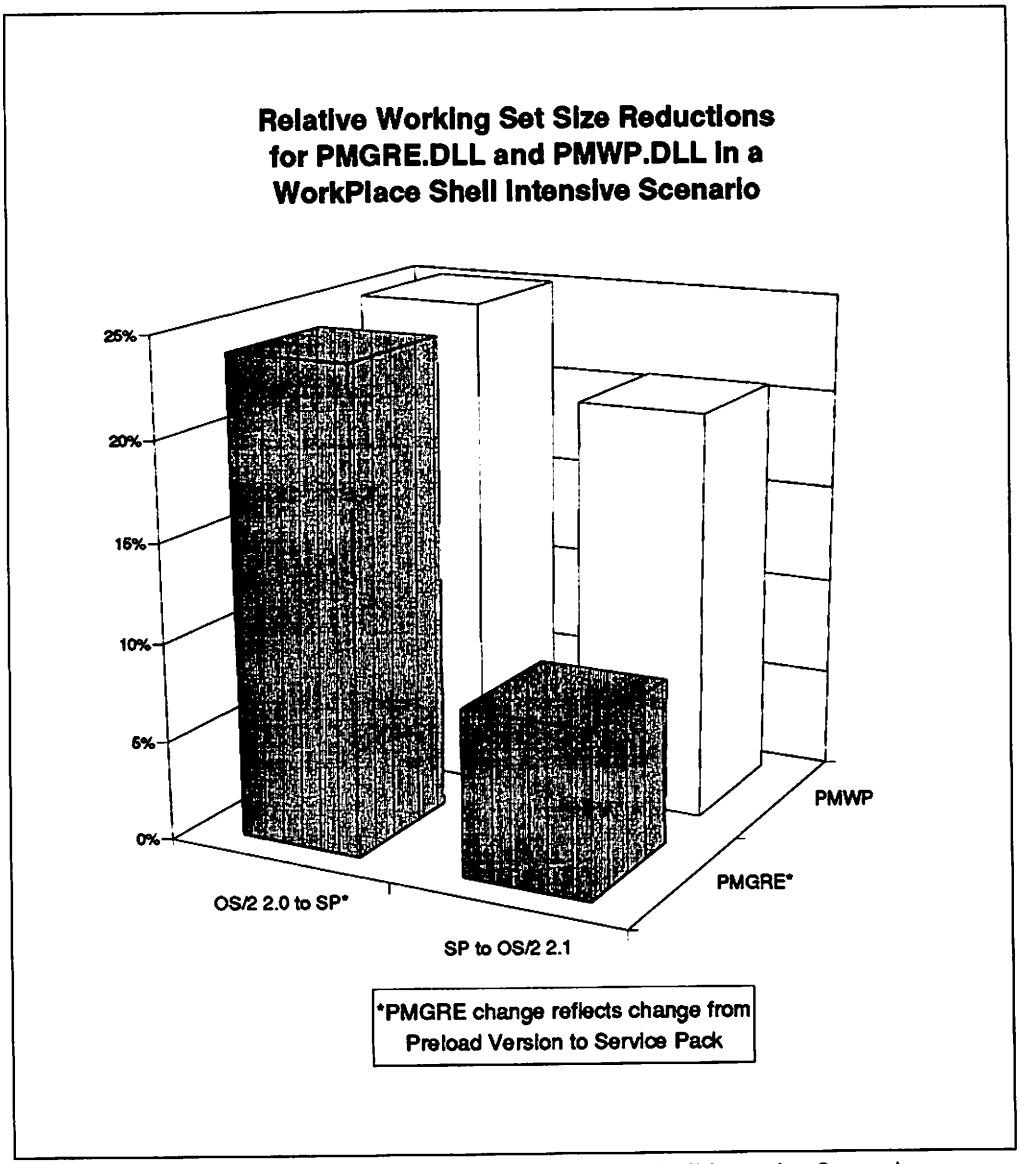


Figure 144. Relative Working Set Reduction: Workplace Shell Intensive Scenario

Figure 144 illustrates the relative working set reductions in a Workplace Shell intensive scenario. This would include manipulating folders, navigating the Workplace Shell, and similar functions. The PMWP.DLL module's working set requirements were reduced 25% in the change from OS/2 2.00.1 to the OS/2 2.0 with Service Pak XR06055 applied, and 20.5% in the change from the Service Pak XR06055 to OS/2 2.1. These are both sizable reductions and are reflected in the increased efficiency of the Workplace Shell. Because many of Workplace Shell functions rely on the graphics engine for processing, there are also considerable reductions in the PMGRE.DLL module. Reductions of 24% and 8%

were achieved in the changes from OS/2 2.00.1 to the Service Pak XR06055 for OS/2 2.0, and from the Service Pak XR06055 to OS/2 2.1, respectively.

The working set of existing functions is becoming smaller and more efficient through the use of page tuning. This should yield benefits for the end user community. However, it should be noted that the benefits of page tuning will not necessarily improve performance in all scenarios. Primary benefits will be seen in environments that are memory constrained or do heavy multitasking. If your working environment has plenty of available memory, reducing the working set will yield less benefit since there is already memory space for applications and system uses.

13.6.3 Functional Enhancements

IBM attempts to enhance an operating system with performance improvements *and* added functionality. These targets, although both beneficial, can also sometimes have conflicting results:

- Reduced working set size through the efforts of page tuning and making OS/2 more efficient
- Increased working set size through new features and functional enhancements

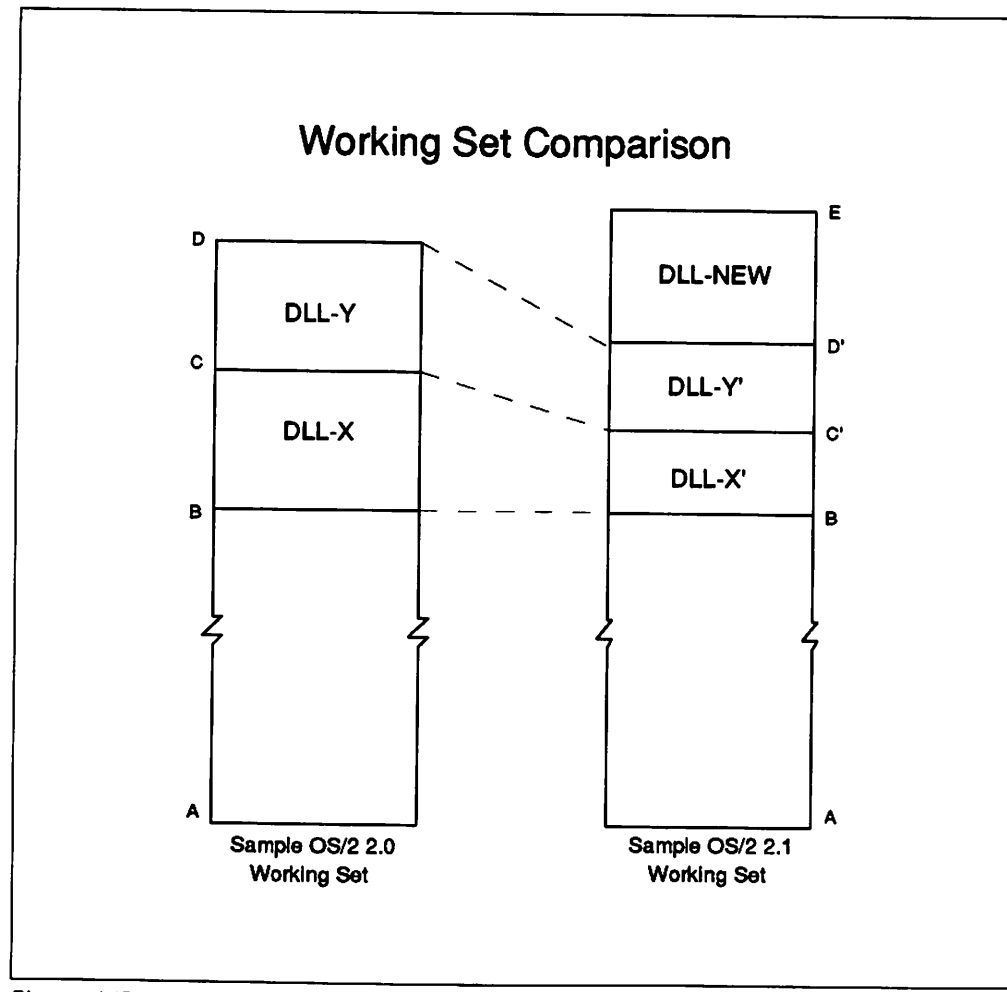


Figure 145. Conceptual Working Set Comparison

IBM has achieved both of these goals, but not without some trade-offs. Although adding function normally increases the working set size, great gains have been made in reducing the working set size of the original product. Before conclusions are reached as to the impact of this, you are reminded that the operating system is gaining efficiency *and* function for only an incremental increase in working set. To understand on a logical level what the "new" working set might look like, please refer to Figure 145 on page 212.

Figure 145 on page 212 illustrates the trade-off of reducing original working set sizes through page tuning versus increasing the working set sizes through adding function. The amount of memory required for memory object DLL-X has been reduced in size as illustrated by DLL-X'. Likewise, DLL-Y has a reduced working set requirement illustrated as DLL-Y'.

It is important to make two points about memory objects (DLLs in this case). First, the actual file system size of DLL-X' or DLL-Y' may not have decreased. Rather, the pages of the DLL's module which are called into memory at any one time may be less. What has changed is how the internal pages of the object are organized. Fewer pages are now required to be committed in memory to perform the similar function as in the past.

Secondly, page tuning yields benefits to the working set by reducing the original working set requirements and making additional space for new features and functions. The reduction of the original working set can be seen by comparing the original working set area of OS/2 2.0, A-D, to the new working set area of OS/2 2.1, A-D' (new function not added in yet). However, the overall size of the new working set is slightly larger than the original working set due to adding function to the operating system. The new function that has been added to the working set is illustrated as DLL-NEW.

13.6.4 Heap Management

The smallest page size (granularity) for memory allocation within a paging environment such as OS/2 is 4KB. If an application allocates memory objects of less than 4KB, a 4KB page is still allocated by the system and there is unusable space (waste) for each of these pages. For one small memory object, this is not very significant. However, most applications allocate large numbers of memory objects. This creates a lot of 4KB pages with very little content and the amount of waste can add up.

There is a type of memory allocation called a "heap" which can aid in reducing this type of waste. The heap is a preallocated amount of memory, in which memory objects less than 4KB can be allocated without the restriction of 4KB boundaries. This eliminates system waste. Heaps can be private or shareable. Heap space is illustrated in Figure 146 on page 214.

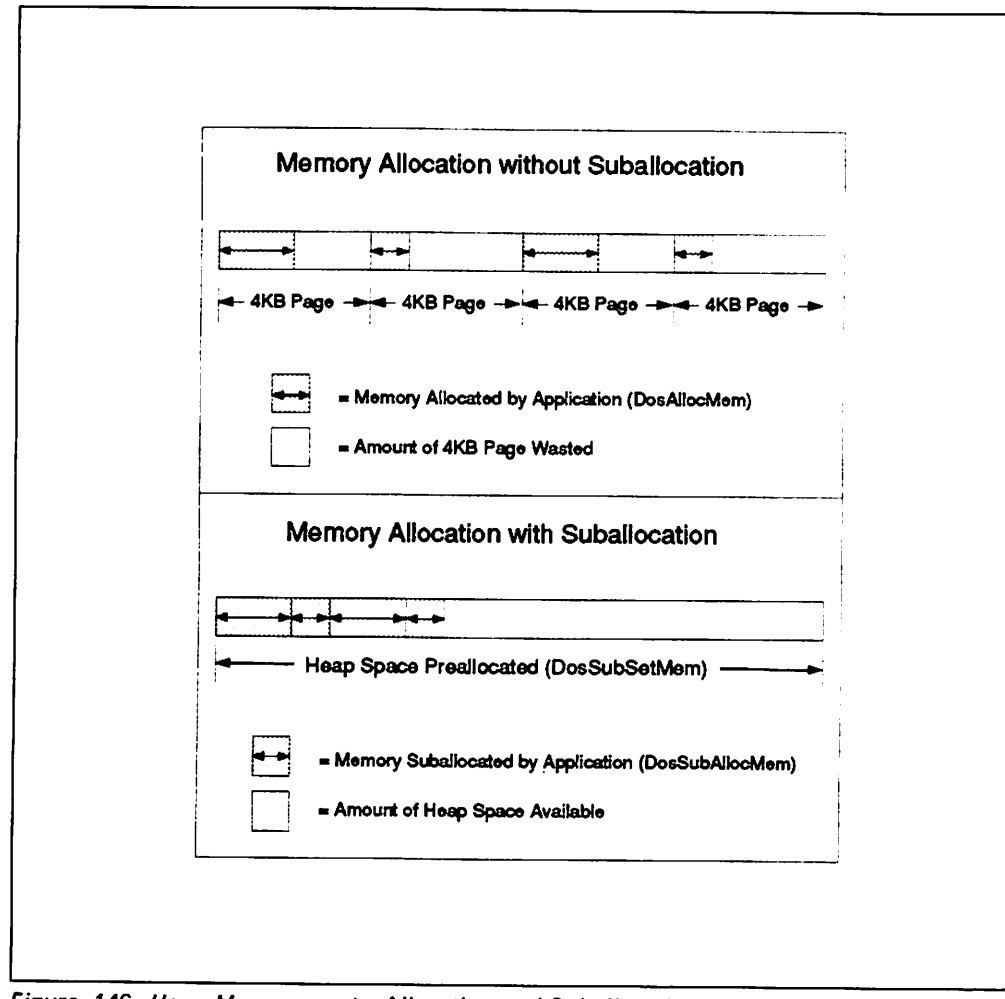


Figure 146. Heap Management - Allocation and Suballocation

Figure 146 shows two memory allocation examples. The top allocation has not been suballocated and therefore has no heap, as opposed to the bottom example which uses a heap. The top memory allocation shows four small memory objects allocated by an application and the waste that is possible without taking advantage of suballocated memory. The heap example shows how these objects can be allocated without regard to the 4KB page boundaries since the heap space was preallocated and its overall space does align on a boundary page.

This subsystem runs via APIs to the DOSCALL1.DLL, system shared library. The suballocation APIs run at the user level (ring 3), similar to the applications requesting them, thereby avoiding constant calls to the kernel APIs (ring 0) and the resources required to make such calls at different privilege levels.

Heap management has been optimized in OS/2 2.1 in how it handles within-heap fragmentation. Memory objects which have been placed in a heap may be discarded or swapped out to disk when they are no longer in use, leaving "holes" in the suballocation range. This needs to be managed efficiently to ensure the heap is effective in its design goal. OS/2 2.1 implements an enhanced algorithm that "cleans up" the heap arena which provides for better heap *coalescence* and *compaction*.

Coalescing refers to the way heaps combine multiple free and contiguous heap spaces into one larger heap space. This coalescing allows for larger suballocations to the heap space by the application requesting memory. Compacting refers to the way heaps "rearrange" the suballocated memory objects within the heap to compact active heap spaces and free larger heap space at the end of the heap area.

Within the heap management structure of OS/2 there are additional facilities for managing how heaps free and maintain suballocations. Figure 147 illustrates how heaps "coalesce" and "compact" to keep heap space optimized.

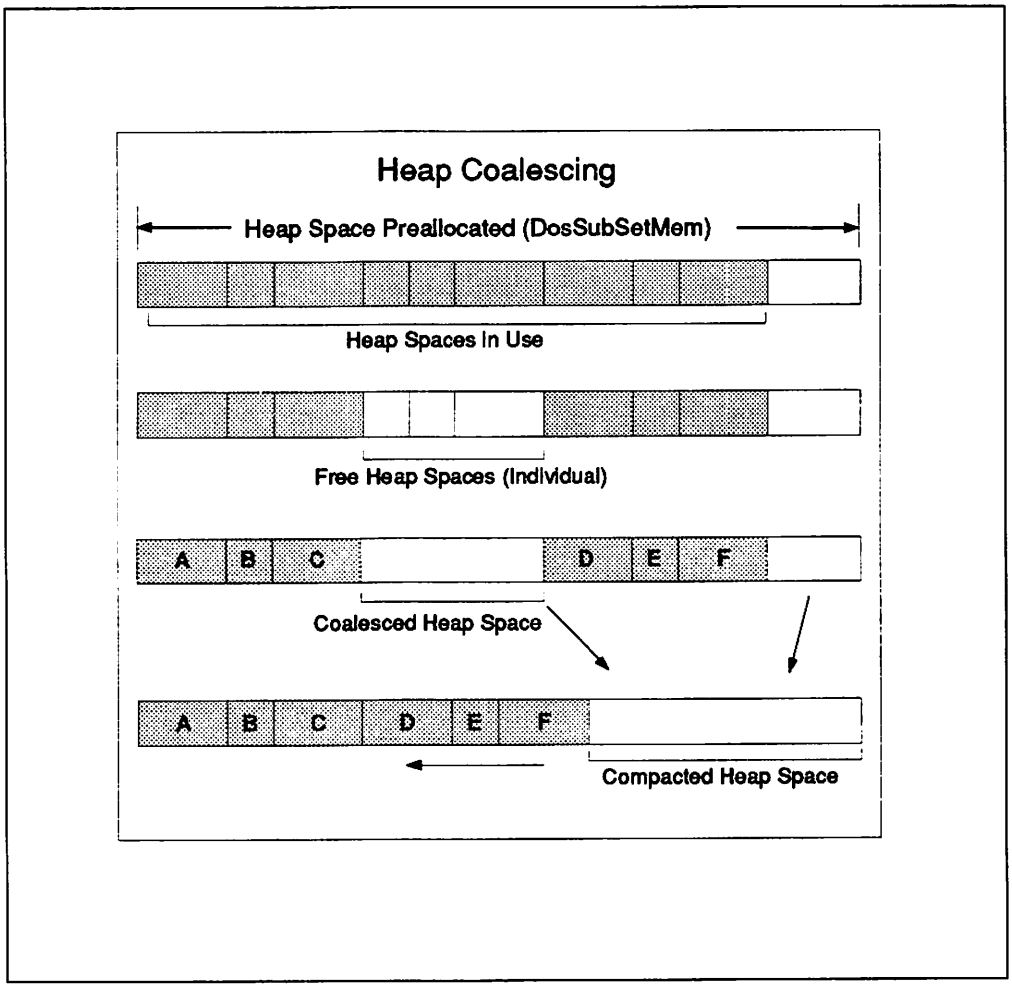


Figure 147. Heap Management - Coalescing and Compacting

Chapter 14. Workplace Shell Programming

Note

This chapter is aimed at the OS/2 programmer, rather than at the OS/2 user or implementer. It assumes that the reader is already familiar with OS/2 and Presentation Manager application programming.

The Workplace Shell provided under OS/2 Version 2.0 introduces an object-oriented layer into the Presentation Manager environment. It provides a mechanism for the registration of object classes, creation of objects within those classes, and the inheritance of characteristics and behaviors from existing object classes. Using the Workplace Shell, an application may be created as a series of objects that interact on the desktop, and which the user manipulates to perform the required application processing. Each object possesses data, which may be defined for the entire class or for each instance, and a set of methods that operate upon that data.

The Workplace Shell functions that allow the creation and manipulation of objects are based upon the **system object model**, which establishes a basic inheritance hierarchy for objects in the system and defines the underlying protocols which regulate the relationships between objects. The concepts behind the system object model are described in detail in *OS/2 Version 2.0 - Volume 3: Presentation Manager and Workplace Shell*, and *System Object Model Guide and Reference*.

This chapter is an enhanced and expanded version of Chapter 7 in *OS/2 Version 2.0 - Volume 4: Application Development, GG24-3774*. It adds more detail about areas of WPS/SOM programming, such as drag/drop and debugging, and provides a sample Workplace Object to illustrate these techniques. This new version of the chapter is included in both the revised version of *OS/2 Version 2.0 - Volume 4: Application Development, GG24-3774*, and also in this document, *OS/2 2.1 Technical Update, GG24-3948*.

14.1 Objects in the Workplace Shell

An object in the Workplace Shell conforms closely to the definition of an application object given in *OS/2 Version 2.0 - Volume 4: Application Development*, in that it consists of a set of data and a number of methods that operate upon that data. Each Workplace Shell object is an instance of a particular object class. In accordance with normal object-oriented theory, the class defines the basic characteristics of the object and the way in which the object responds to events.

14.1.1 Inheritance Hierarchy

Each object class is descended from another class, known as its **parent class**. Since the system object model supports the object-oriented concept of inheritance, a class may inherit data and methods from its parent class, which in turn may inherit data and methods from *its* parent, and so on. A class which inherits properties from other classes is therefore known as a **descendant** of those classes, and the classes from which it inherits are known as **ancestors**. The implementation of inheritance in the Workplace Shell means that when

creating a new object class, a programmer simply subclasses the parent class, and need only define those characteristics that are not defined by, or are different from those of the parent class. This greatly simplifies the process of creating a new object class.

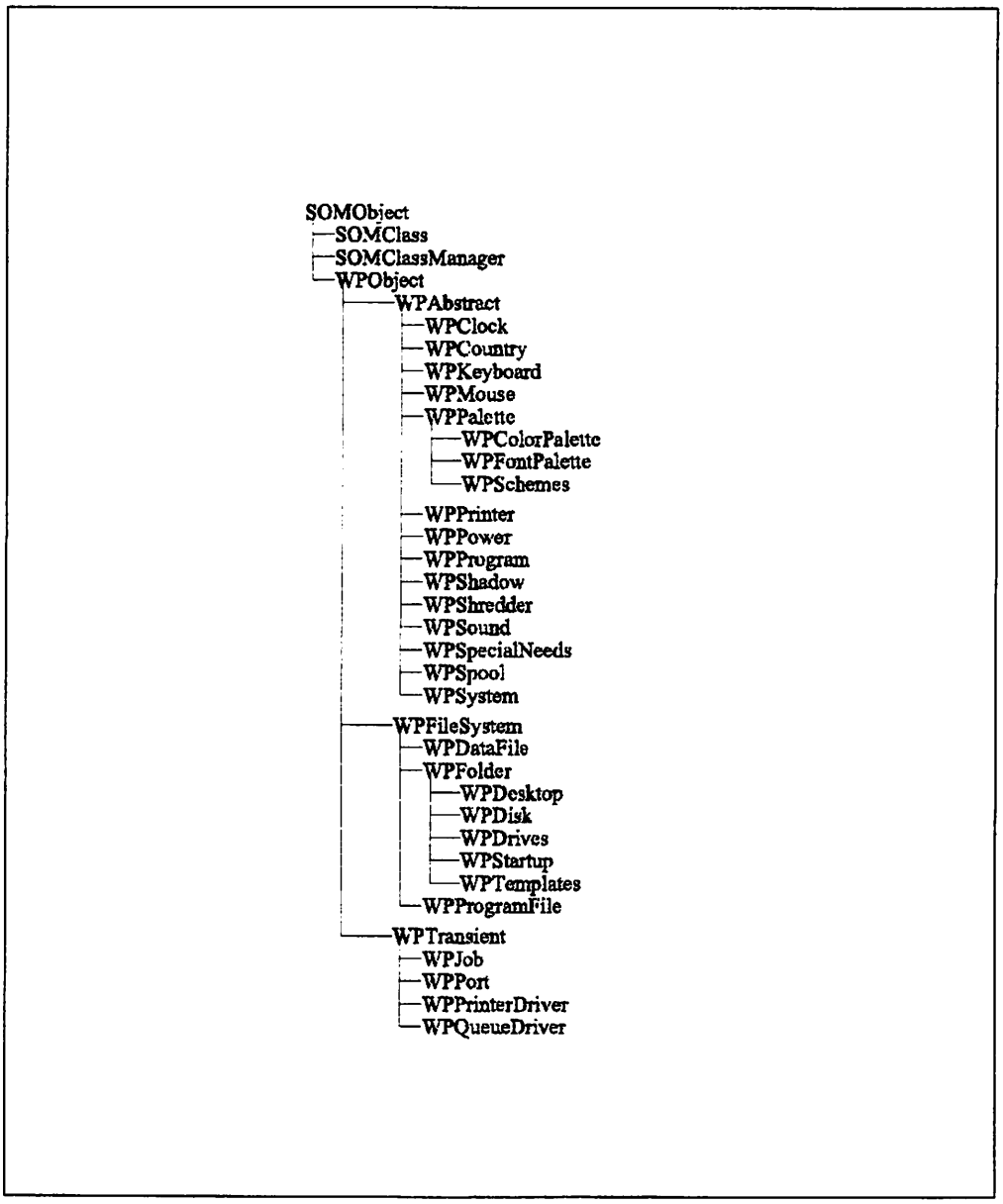


Figure 148. Workplace Shell Inheritance Hierarchy

Under the system object model, every object class is a descendant of the base class *SOMObject*. This class defines the basic characteristics and behaviors common to all objects in the system. Other object classes are subclasses of this class. The system object model provides two additional classes, *SOMClass* and *SOMClassManager*, to form the basis of an inheritance hierarchy. The Workplace Shell extends this hierarchy by creating a number of classes of its own, based upon the *SOMObject* class. These Workplace Shell object classes define the characteristics of the object types that are defined and implemented by the Workplace Shell itself.

The inheritance hierarchy implemented by the Workplace Shell is illustrated in Figure 148.

As well as being descended from the system object model base inheritance hierarchy, all Workplace Shell object classes are descended from one of three **base storage classes** defined by the Workplace Shell. These classes are so named because they directly influence the storage of control information and instance data for the class. The three predefined base storage classes are:

- **WPAbstract**, which is the base class for abstract objects such as programs, devices, etc., and for which control information is stored in the system initialization file OS2.INI.
- **WPFileSystem**, which is the base class for objects that are stored as files in the file system, and for which control information is stored in the file system as extended attributes.
- **WPTransient**, which is the base class for objects that only exist during execution of a particular program; that is, the object is created and used for a particular purpose during processing, and then immediately deleted from the system.

An application developer may extend the Workplace Shell inheritance hierarchy by introducing new object classes based upon those already implemented by the Workplace Shell itself. Indeed, the developer may even introduce new base classes, although this is definitely a non-trivial exercise and should be approached with caution.

14.1.2 Metaclasses

Just as each Workplace Shell object is an instance of a class, the class itself is an instance of another class known as its **metaclass**. Just as an object has instance data and methods that pertain only to a specific instance of the class, so the metaclass has class data and methods that pertain to the *entire* class. Such methods are known as **class methods**, whereas methods that operate only for a particular instance of the class are known as **instance methods**.

Class methods and data are available to the programmer when creating new object classes. A programmer may introduce new class data and methods for an object class, as well as instance data and methods. Similarly, a new object class may override existing class methods to modify the processing performed by those methods.

14.1.3 Class Implementation

Each object class in the Workplace Shell resides in a dynamic link library (DLL). A programmer creates an object class by defining its characteristics in a **class definition file**. This file is then used as input to the **SOM Precompiler**, in order to produce "C" source code and header files for the object class. This source code includes basic definitions for the object class's data and methods; the code is then edited by the programmer to include the logic for each of the required methods. Once the code is complete, it is compiled and link edited in the normal way to produce a dynamic link library; see OS/2 2.1 Volume 4: Writing Applications, Chapter 14 "Compiling and Link Editing an Application" for further information on compiling and link editing.

When an object class has been created, it must be registered with the Workplace Shell, which includes the DLL in a list of libraries loaded at initialization time.

The entry points for the DLL are known to the Workplace Shell, and may be called in order to invoke the object's methods.

The process of creating an object class from a class definition file is described in 14.3, "Defining an Object" on page 230.

14.2 Object Structure

In the simplest case, an object in the Workplace Shell consists of methods and instance data. The PM Window Manager communicates events to the Workplace Shell using messages, which in turn invokes the object's methods to perform the processing indicated by the event. This is in accordance with the definition of an application object given in Chapter 4, "The Presentation Manager Application Model", of *OS/2 2.1 Volume 4: Writing Applications*. Note that since the Workplace Shell provides a more extensive inheritance hierarchy than the base Presentation Manager application model, the method invoked by a particular message may belong explicitly to the object in question, or may belong to its parent (and be inherited from that parent).

The structure of an object in the Workplace Shell is therefore very similar to that of a window in the conventional Presentation Manager application model; the Workplace Shell object simply takes the object-oriented concepts to a higher degree of implementation. Therefore the constructs implemented by Presentation Manager under previous versions of OS/2 can often be implemented more elegantly with the Workplace Shell.

For the remainder of this chapter, two examples are used to explain the structure and behavior of an object class. These are the pwFolder and pwFinanceFile Workplace Objects.

- The pwFolder Workplace Object is a specific type of Workplace Shell folder which has a password defined so that it can be locked to prevent access by an unauthorized user. This object class is implemented by subclassing the *WPFolder* class to create a new object class named *PWFolder*, adding new methods and overriding existing methods where appropriate.
- The pwFinanceFile Workplace Object is a specific type of Workplace Shell data file which has a password defined so that it can be locked to prevent access by an unauthorized user. Additional methods have been added to provide specific behavior and this is covered later in this chapter. This object class is implemented by subclassing the *WPDataFile* class to create a new object class named *PWFinanceFile*, adding new methods and overriding existing methods where appropriate.

Sample code is provided in the text for the various methods used to add the password protection to the folder.

14.2.1 Methods

In a Presentation Manager application, a window procedure receives messages from Presentation Manager, determines the type of message and invokes a series of program statements (which effectively constitute a method) as a result of that message. A Workplace Shell object operates in a similar fashion, except that the Workplace Shell itself determines the type of message and invokes the corresponding method, without any explicit action on the part of the object.

Therefore, whereas the Presentation Manager window procedure comprises a *case* statement with each case being a method, the Workplace Shell object eliminates the need for the *case* statement and allows the Workplace Shell to invoke the methods directly. The syntax for invoking a method from within an object or application is hence very similar to that for invoking a subroutine; the only real difference is that a method may be accessed from outside the object itself (that is, from another object or from an application), while a subroutine is normally private to the object.

Many methods are defined by the *WPObject* class, from which application-defined classes are typically descended. When creating a new object class, a programmer may override the methods already defined by the class's ancestors, and/or include new methods specific to the class being created. The methods defined by the *WPObject* class are described in the *IBM OS/2 Version 2.0 Presentation Manager Reference*. Programmers who wish to create new object classes descended from this case should read the descriptions of these methods to determine the extent of the modifications necessary.

14.2.1.1 Invoking a Method

As mentioned in Chapter 4 of *OS/2 2.1 Volume 4: Writing Applications*, methods within an object are invoked as a result of messages that communicate events to the object. These events may be initiated by the user (for example, as a result of clicking the mouse on an object's context menu), by the object itself or another object, or by the system to indicate a system event such as opening or closing a view of the object.

The syntax for invoking a method is similar to that for invoking a subroutine, with one exception. The first parameter passed in the call is a pointer to an object that is capable of invoking the method called the "receiver", and this is typically a pointer to the object itself. This is illustrated in Figure 149, where a sample invocation of a method named _wpSetTitle is shown.

```
PWFolder *somSelf;                          /* Pointer to self      */
PSZ      szTitle;                           /* Title string         */

_wpSetTitle(somSelf,szTitle);               /* Set title string     */
```

Figure 149. Invoking a Method

The _wpSetTitle method is defined by the *WPObject* class, and is inherited by all classes descended from the class. The method accepts a title string and sets the title of the object, that is, the text that appears below the object's icon on the Workplace Shell desktop.

The pointer *somSelf* is defined by the SOM Precompiler when it creates the "C" source code from the class definition file. In the example above, *somSelf* is defined as a pointer to an object of class *PWFolder* and, within a method, allows the method to access the instance data of the object to which it belongs. The need to pass this pointer arises from the limitations of the "C" language syntax under which the current implementation of the Workplace Shell operates; other languages such as C++ may be able to invoke methods in a more elegant manner.

14.2.1.2 Method Processing and Instance Data

Within a method, the *somSelf* pointer, passed as the first parameter in the call to the method, acts as a pointer to the method's own object, and allows the method to access its instance data. The SOM Precompiler automatically provides a base pointer named *somThis* that references the instance data, and includes a call to a method that initializes this pointer from the object pointer:

```
PWFolderData *somThis = PWFolderGetData(somSelf);
```

When this statement has successfully executed upon entry to the method, the method has access to the object's instance data. For example, the password-protected folder has a password string, which may be accessed by a method using the following name:

```
somThis->szPassword
```

To make things simpler, the SOM Precompiler generates a macro for each instance variable, in a manner similar to that used for function names:

```
#define _szPassword (somThis->szPassword)
#define _szCurrentPassword (somThis->szCurrentPassword)
#define _szUserid (somThis->szUserid)
```

This macro is included in a header file for the object class, and avoids the need for the programmer to type the complete name throughout the source code.

Once the instance data is available to the method, any application logic may be performed, including the use of OS/2 and Presentation Manager resources. See 14.4.5, "Accessing PM Resources From a Workplace Shell Object" on page 265 for additional considerations on the use of Presentation Manager resources from within a Workplace Shell object.

14.2.1.3 Returning from a Method

In order to return control to its calling routine, a method simply uses the *return* statement. Any valid form of return code may be passed to the calling routine as a parameter to this statement, provided that the data type of the return code is consistent with the declaration of the method. The data type of the return code is typically set by the SOM Precompiler, and a default *return* statement provided, based on information supplied by the programmer when the method is defined in the *Methods* section of the class definition file (see 14.3.2, "Class Definition File" on page 231).

14.2.1.4 Overriding Existing Methods

A new object class may override one or more of the existing methods defined by its parent class, either to completely replace the processing performed by these methods, or to add its own processing to that already performed by the parent. An example of an object class overriding the _wpSetTitle method is shown in Figure 150 on page 223.

```
SOM_Scope BOOL SOMLINK pwfolder_wpSetTitle(PWFolder *somSelf,
                                           PSZ pszNewTitle)
{
    CHAR szBuf[100];                              /* Character buffer      */

    PWFolderData *somThis =                       /* Get instance data     */
                 PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",               /* Set debug info        */
                        "pwfolder_wpSetTitle");

    strcpy(szBuf,pszNewTitle);                    /* Get current title     */

    if ((strcmp(_szCurrentPassword,               /* If folder is locked   */
                _szPassword)) != 0)
        if((strstr(szBuf,"LOCKED")) == NULL )     /* and <LOCKED> not in   */
                                                  /* current title         */
          strcat(szBuf," <LOCKED>");              /* Add <LOCKED> to title */

    return(parent_wpSetTitle(somSelf,             /* Allow default proc to */
                             szBuf));             /* occur                 */
}
```

Figure 150. Overriding an Existing Method. This example shows the _wpSetTitle method being overridden to add the word "LOCKED" to the end of the title of a locked password-protected folder.

The example given in Figure 150 shows the use of class-specific processing to modify the title of a password-protected folder. The inclusion of the string "<LOCKED>" at the end of the user-specified title provides a visual indication to the user that the folder is locked. Additional visual indication is provided by modifying the icon when the folder is in the locked state; the code that carries out this operation is included in the *_LockFolder* method shown in Figure 151 on page 224.

The strings *_szCurrentPassword* and *_szPassword* are instance data items defined by the new object class. These data items are actually accessed using the *somThis* pointer. However, the SOM Precompiler defines a macro for each instance data item, as described in 14.2.1.2, "Method Processing and Instance Data" on page 222.

Note that most workplace methods require that parent processing be performed during the override function. Normally this would be part of the return statement, but some methods require parent processing to be done first. You should check the method description to determine where the parent processing needs to be done.

14.2.1.5 Adding New Methods

In addition to overriding existing methods defined by the parent class, an object class may also add new methods to carry out processing for events not handled by the parent class. For example, the password-protected folder example must have a mechanism to lock the folder. This is implemented as a new method named *_LockFolder*, as shown in Figure 151 on page 224.

```
SOM_Scope BOOL SOMLINK pwfolder_LockFolder(PWFolder *somSelf)
{
    HPTR  hLockedIcon;

    PWFolderData *somThis =                    /* Get instance data     */
                PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",            /* Set debug info        */
                        "pwfolder_LockFolder");

    strcpy(_szCurrentPassword,                 /* Invalidate current    */
           "NOPASSWD");                        /* password              */
    _wpSetTitle(somSelf,                       /* Set title             */
                _wpQueryTitle(somSelf) );

    hLockedIcon = WinLoadPointer(HWND_DESKTOP, /* Load "lock" icon      */
                                 (HMODULE)0,
                                 LOCK);

    _wpSetIcon(somSelf,                        /* Set icon to locked    */
               hLockedIcon);                   /* appearance            */

    return((BOOL)0);                           /* Return                */
}
```

Figure 151. Adding a New Method

This method simply copies a default string to the variable *_szCurrentPassword* that contains the last supplied password entry from the user, so that when a comparison is made between this variable and the folder's password, the two do not match. This effectively locks the folder and prevents any view of it being opened. To provide a visual indication to the end user that the folder is locked, a "locked" icon is loaded using the Presentation Manager **WinLoadPointer()** function, and the _wpSetIcon method is invoked to set this as the folder's new icon on the desktop.

Note that the definition for adding a new method is very similar to that for overriding an existing method. The primary difference is that, since the new method is specific to the object class and is not defined by the parent class, there is no need to invoke the parent class's method to perform default processing for the method.

14.2.1.6 Attaching a Method to the Context Menu

A method may be invoked as a result of the user selecting an item from the object's context menu. In order to allow this, an item must be added to the context menu, and an appropriate action must be taken by the object when that item is selected by the user.

An item can be added to the context menu for an object class by overriding the _wpModifyPopupMenu method defined by the *WPObject* class, and including a call to the _wpInsertPopupMenuItem method to insert the item. This technique is shown in Figure 152 on page 225.

```
#define MI_LOCK     WPMENUID_USER+1
   :
   :
SOM_Scope BOOL SOMLINK pwfolder_wpModifyPopupMenu(PWFolder *somSelf,
                                                  HWND hwndMenu,
                                                  HWND hwndCnr,
                                                  ULONG iPosition)
{
    PWFolderData *somThis =                      /* Get instance data     */
                 PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",              /* Set debug info        */
                        "pwfolder_wpModifyPopupMenu");

    _wpInsertPopupMenuItems(somSelf,             /* Insert menu item      */
                            hwndMenu,            /* Menu handle           */
                            iPosition,           /* Default position      */
                            hModule,             /* Module handle         */
                            MI_LOCK,             /* Menu item id          */
                            0);                  /* No submenu id         */

    return(parent_wpModifyPopupMenu(somSelf,     /* Allow default proc to */
                                    hwndMenu,    /* occur                 */
                                    hwndCnr,
                                    iPosition));
}
```

Figure 152. Adding an Item to a Context Menu

The example shown in Figure 152 adds a *Lock* item to the context menu for the password-protected folder object. This allows the folder to be locked by the user at any time, regardless of whether a view of the folder is currently open.

The _wpInsertPopupMenuItem method adds a menu item or a submenu to the existing context menu for the object. The item identifier for the menu item or submenu (MI_LOCK in the above example) is an integer constant that is typically defined in the header file. Note that the value of this constant should be specified as an offset from the system-defined constant WPMENUID_USER, rather than an absolute integer value. Following this convention will avoid any clashes with item identifiers defined by the Workplace Shell for default context menu items.

Since the password-protected folder is a descendant of the *WPFolder* class defined by the Workplace Shell, the default context menu items for the *WPFolder* class should also appear. The default processing for the parent class is therefore invoked as part of the _wpModifyPopupMenu processing for the new object class.

Once the required item is added to the context menu, the object must be able to detect when the item is selected in order to invoke the appropriate method. By default, the _wpMenuItemSelected method is invoked by the system whenever the user selects an item from the context menu. This method, which is defined by the *WPObject* class, may be overridden by a new object class in order to check for the presence of a new item and invoke the appropriate method. The item identifier of the selected item is passed as a parameter to the _wpMenuItemSelected method, and is normally interrogated using a *case* statement, as shown in Figure 153 on page 226.

```
SOM_Scope void SOMLINK pwfolder_wpMenuItemSelected(PWFolder *somSelf,
                                                   HWND hwndFrame,
                                                   ULONG MenuId)
{
    PWFolderData *somThis =                        /* Get instance data     */
                PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",                /* Set debug info        */
                        "pwfolder_wpMenuItemSelected");

    switch (MenuId)                                /* Switch on item id     */
           {
           case MI_LOCK:                           /* If "Lock" item        */
                _LockFolder(somSelf);              /* Lock folder           */
                break;
           default:                                    /* else              */
                parent_wpMenuItemSelected(somSelf,     /* Allow default     */
                                          hwndFrame,   /* processing to     */
                                          MenuId);     /* occur             */
                break;
           }
    return;
}
```

Figure 153. Invoking a Method via a Context Menu Item

The _wpMenuItemSelected method consists of a *case* statement that determines the item selected from the context menu. In the above example, an explicit case is included only for the MI_LOCK item defined by this class. All other menu items are defined by the parent class, and their selection is therefore handled by allowing the parent class's default processing to occur.

14.2.1.7 Modifying the Standard Context Menu Items

The _wpFilterPopupMenu method can be used to filter out (remove) standard menu items that are inherited from the ancestor classes, or to reinstate any of the standard pop-up menu items. This method can also be used to determine if the ancestor classes have filtered out any of the Workplace Shell-provided standard menu items, by checking to see if any of the flags associated with the menu items are not set.

The *ulFlags* parameter of "C" type *ULONG* is really a bit array which is binary *OR*ed together with the ancestor classes *ulFlags* when the parent method is called, effectively adding these menu items together. The resultant *ulFlags* is then returned from the _wpFilterPopupMenu method.

But if the parent method is called first, then the resultant flags are binary *AND*ed with the complement of the menu item (flag) to be removed. Upon returning this from the object's _wpFilterPopupMenu, the item will now be removed from the pop-up menu.

To determine if a menu item is present or not, first call the parent method and then simply binary *AND* the menu item flag with the parent method result. If the result of this operation is the menu item flag that was *AND*ed, then the flag has been set by the ancestor classes; otherwise it has been removed.

Figure 154 on page 227 shows how to test for a menu item, removing the menu item if it is present, or adding it if the ancestor classes removed it. In this case the *Create another* menu item is the menu item of interest.

```
SOM_Scope ULONG
          SOMLINK pwFinanceFile_wpFilterPopupMenu(PWFinanceFile *somSelf,
         ULONG ulFlags,
         HWND hwndCnr,
         BOOL fMultiSelect)
{ ULONG ulPopupFlags;

    PWFinanceFileData *somThis = PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile",
                                        "pwFinanceFile_wpFilterPopupMenu");

    /* first find out what our ancestors have done! */
    ulPopupFlags = parent_wpFilterPopupMenu(somSelf,ulFlags,
                                        hwndCnr,fMultiSelect);

    /* now what has been done to the "Create another" menu item */
    if ((ulPopupFlags & CTXT_NEW) == CTXT_NEW) {

       /* the "Create another" menu item is on our Popup, so remove it */
       ulPopupFlags = ulPopupFlags & ~CTXT_NEW;
    } else {

       /* the "Create another" menu item is NOT on our Popup, so add it */
       ulPopupFlags = ulPopupFlags | CTXT_NEW;
    } /* endif */

    return(ulPopupFlags);
}
```

Figure 154. Filtering the Pop-Up Menu Items

14.2.1.8 Class Methods

Most object methods are instance methods; that is, they act upon one particular instance of an object class, rather than upon all instances of the class.

However, there are times when it is useful to have methods that operate on the object class itself. These methods may operate on class data rather than instance data, thereby affecting the entire class rather than a single instance of the class. Such methods are known as class methods. The class method _wpclsQueryTitle is defined by the *WPObject* class, and is overridden in the password-protected folder example. An example of the overridden _wpclsQueryTitle method is given in Figure 155 on page 228.

```
PSZ szDefaultClassTitle = "Password Folder";

/*
 *
 *  METHOD: wpclsQueryTitle                                    PUBLIC
 *
 *  PURPOSE:
 *    Return the string "Password Folder"
 *
 */
#undef SOM_CurrentClass
#define SOM_CurrentClass M_PWFolderCClassData.parentMtab
SOM_Scope PSZ SOMLINK pwfoldercls_wpclsQueryTitle(M_PWFolder *somSelf)
{
    /* M_PWFolderData *somThis = M_PWFolderGetData(somSelf); */
    M_PWFolderMethodDebug("M_PWFolder","pwfoldercls_wpclsQueryTitle");

    return(szDefaultClassTitle);
}
```

Figure 155. Class Method Example. This example shows an overridden class method _wpclsQueryTitle, which is modified to supply a default title for an object within the class.

The purpose of this class method is to provide the password-protected folder with a default title. This is the title that will appear with the folder's template icon in the *Templates* folder, and which is given to any instances of the class that are instantiated without a title. Since the default title applies to all instances of the class, it is implemented in a class method rather than an instance method.

The prefix "M_" denotes the metaclass in the SOM-generated "C" source. As already mentioned, the first parameter passed to a method is a pointer to a type of object that can invoke that method; this is true for both instance methods and class methods; for a class method the first parameter contains a pointer to an instance of the metaclass.

```
          Pointer to instance of metaclass
          which is a class object ──────┐
                                        ▼
pwfoldercls_wpclsQueryTitle(M_PWFolder *somSelf)
                            ▲
     Type is Metaclass ─────┘
```

Since a class is also an object, it follows that the class itself has its own "instance data"; hence the next line of code appears as follows:

```
/* M_PWFolderData *somThis = M_PWFolderGetData(somSelf); */
```

This statement would access the SOM object's class data. However, since no class data is specified in the .CSC file, there is nothing to access and so the SOM Precompiler has commented the line out to reflect this.

For simple examples, it is easier to use global variables in the DLL for class data. This technique has been used in Figure 155; the default title string is stored at the beginning of the program into the global variable *szDefaultTitle*.

However, using this technique means that class data can be accessed by instance methods, which is never desirable, and may have adverse consequences, although these may generally be avoided by sound programming techniques.

14.2.1.9 Invoking Another Object's Methods

An object may invoke a method in another object class. This technique is useful in a client-server situation, where one object creates another object of a different class and then wishes to have that object perform certain actions. The system object model provides programming functions that can be used to determine the necessary information and invoke the method. An example is given in Figure 156.

```
SOMAny *RecordClass;                              /* Class object pointer  */
somID  idQueryMethod;                             /* Method id             */

CHAR   szQueryBuffer[100];                        /* Query data buffer     */
PVOID  pFindData;                                 /* Returned data buffer  */
   :
   :
rc = DosAllocSharedMem(&pFindData,                /* Alloc shared memory   */
                       NULL,                      /* No name               */
                       sizeof(szQueryBuffer)+1,   /* Size of memory object */
                       OBJ_GIVEABLE  |            /* Make object giveable  */
                       PAG_WRITE     |            /* Allow write access    */
                       PAG_READ      |            /* Allow read access     */
                       PAG_COMMIT);               /* Commit storage now    */

strcpy(pFindData,szQueryBuffer);                  /* Copy data to buffer   */

RecordClass = _somFindClass(SOMClassMgrObject,    /* Get class obj pointer */
                            SOM_IdFromString("Record"),
                                                 1,1));
idQueryMethod = SOM_IdFromString("clsQuery");     /* Get method id         */

_somDispatchL(RecordClass,                        /* Invoke method         */
              idQueryMethod,                      /* Method id             */
              (void *)0,                          /* No descriptor string  */
              pFindData,                          /* Method parameters     */
              somSelf);
```

Figure 156. Invoking a Method in Another Object Class

The example given in Figure 156 shows part of a "database client" object that sends a database query to a "database server" object. The client first allocates a shared memory object into which it loads the query. The client then uses the *_somFindClass* method and the *SOM_IdFromString* macro to determine the object pointer for the object, and the method identifier for the required method. The *_somDispatchL* method is then used to invoke the method.

It is also possible to invoke a class method using the object pointer to that class, obtained using the *_somFindClass* method shown in Figure 156. This requires the header file for the class to be included in the source code for the class that will invoke the method, using a *#include* statement. In the module definition file for the invoking class, the following IMPORT statements must be provided:

```
IMPORTS
   record.RecordCClassData
   record.RecordClassData
   record.RecordNewClass
   record.M_RecordCClassData
   record.M_RecordClassData
   record.M_RecordNewClass
```

When these steps have been carried out, a method in the other class may be invoked directly, as follows:

```
_clsQueryDatabase(RecordClass,          /* Invoke class method   */
                  pQuery,               /* Method specific       */
                  Folder);              /* parameters            */
```

Although this technique is less clean than the previous approach since it requires the inclusion of the header file and import statements, it provides better performance.

14.2.2 Subroutines

Subroutines may be accessed from within a Workplace Shell object, in much the same manner as from any other program. Normal programming language calling conventions are used. Subroutines used by the object may reside within the same DLL as the object itself, or may be in a different DLL.

A number of guidelines for the use of subroutines within Presentation Manager applications are given in Chapter 4 of *OS/2 2.1 Volume 4: Writing Applications*. Note that similar guidelines apply to the use of subroutines within Workplace Shell objects, since these objects should also adhere to object-oriented programming principles.

14.3 Defining an Object

The definition of an object is achieved using a language known as the **Object Interface Definition Language**. The statements that define an object class are entered into the class definition file for the class, which is an ASCII file and may thus be created using any normal text editor. The class definition file is used as input to the SOM Precompiler, which will generate a number of files from the class definition file.

14.3.1 Files

The SOM Precompiler generates a number of files that are used to define an object class to the Workplace Shell and to other classes that may wish to inherit the characteristics and behaviors of the class. These files are:

.H A public header file for programs that use the class.

.PH A private header file, which provides usage bindings to any private methods implemented by the class.

.IH An implementation header file, which provides macros, etc., to support the implementation of the class.

.C A template C file, to which code may be added to implement the class.

.SC A language-neutral class definition.

.PSC A private language-neutral core file, which contains private parts of the interface for the class.

.DEF An OS/2 DLL module definition file containing the relevant exports need to implement the class.

These files may then be used as input to a C compiler, generating object code that is in turn linked to create a dynamic link library, which implements the object class.

14.3.2 Class Definition File

The class definition file contains all the information necessary to implement a new class. The file is divided into the following sections:

1. Include section
2. Class section
3. Parent Class section
4. Release Order section
5. Metaclass section
6. Passthru section
7. Data section
8. Methods section

Each of these sections is described in more detail below, using examples from the password-protected folder class described earlier in this chapter.

14.3.2.1 Include Section

Since all system object model classes have a parent, it is necessary to know the name of the parent class and the location of its interface definition. The include section specifies the location of the interface definition file for the parent. In the folder example, only a single line is included:

```
#
# Include the class definition file for the parent class
#
include <wpfolder.sc>
```

Since the folder example is simply a specialized form of the *WPFolder* class, it uses this class as its parent and inherits much of its behavior from the *WPFolder* class. The include section therefore specifies the interface definition for the *WPFolder* class. A full list of Workplace Shell classes and their definition files can be found in the *IBM OS/2 Version 2.0 Presentation Manager Reference*.

Note that the comments that start with a "#" are discarded by the SOM Precompiler; hence the comment in the example above will not be seen in the SOM Precompiler-generated files.

14.3.2.2 Class Section

This section provides basic information about the new class, specifying its name and various attributes. The password folder example has the following class section entry:

```
#
#   Define the new class
#
class: PWFolder,
       file stem = pwfolder,
       external prefix = pwFolder_,
       class prefix = pwFoldercls_,
       major version = 1,
       minor version = 1,
       local;
-- PWFolder is a Password-protected folder.
-- Its derived as follows:
--        SOMObject
--          - WPObject
--             - WPFileSystem
--                - WPFolder
--                   - PWFolder
```

All class definition files must contain a class section. Certain statements within the class section are mandatory, while others are optional.

The first item in the class section is a *name*:

```
class: PWFolder,
```

All classes must have a name.

The *file stem* specifies the file name to be used by the SOM Precompiler for the generated files. For example, if the file stem statement reads:

```
file stem = myfile
```

then the .DEF file generated by the SOM Precompiler would be called *myfile.def*.

The *external prefix* specifies a prefix to be used by the SOM Precompiler on all function names. Hence if an external prefix of "pwFolder_" is specified and a method is named "SetInfo," the function name generated by the SOM Precompiler would be "pwFolder_SetInfo."

The SOM Precompiler normally generates a macro for all methods defined by the class, such that the method is referenced in the source code by its defined name, preceded by an underscore character. For example, the method *pwFolder_SetInfo* described above would be referenced simply as *_SetInfo*. This helps make the source code more readable and avoids the need for the programmer to type the full name when editing the code.

The *class prefix* is similar to the external prefix, except that it is used specifically for functions that are class methods. The differences between class methods and instance methods are discussed in 14.2.1.8, "Class Methods" on page 227.

The *major version* and *minor version* are used to ensure that the bindings are at the right level for the class implementation code.

The *local* option is used to specify that binding files should be linked locally. In "C" programming terms, this means that the following source code is generated:

```
#include "wpfolder.h"
```

If the *global* option is used, the resulting source code would be as follows:

```
#include <wpfolder.h>
```

The last part of the class section is for comments. Using "--" as the comment style causes a comment block to be passed through to the interface definition (.SC) file.

14.3.2.3 Parent Class Section

The parent class section specifies the parent of the new class. All classes must have this section. The parent class section for the password-protected folder example appears as follows:

```
#
# Parent class
#
parent: WPFolder;
```

14.3.2.4 Release Order Section

This section allows the programmer to specify the sequence in which the methods and public data will be released. Since this sequence is maintained by the SOM Precompiler, other programs using this class will not need to be recompiled every time something new is added to the class.

Note that for future compatibility it is essential that all public and private methods are listed in the release order section, and their order does not change. It is strongly suggested that the "-r" option be used with the SOM compiler to produce any release order warnings.

The password-protected folder example has only one public method in addition to those already defined by its ancestor classes. This method is seen in the release section as follows:

```
#
# Specify the release order of new methods
#
release order: LockFolder;
```

Since other public methods are defined by the parent class or by its ancestors, the programmer creating an object class need not define these methods in the class definition file. Hence the programmer need not be aware of the existing methods in the parent class, unless they require modification for use by the new class. This is in accordance with the object-oriented concept of encapsulation.

14.3.2.5 Metaclass Section

For the password-protected folder example (and in most other cases) an explicit metaclass is not required. The concept of metaclasses is discussed in 14.1.2, "Metaclasses" on page 219. Readers desiring more knowledge of programming using metaclasses should refer to the *IBM SOM Programming Reference*.

14.3.2.6 Passthru Section

This section allows the programmer to define blocks of C source code that are passed through to any of the files generated by the SOM Precompiler. Each passthru block is distinguished by an identifier, the syntax of which is as follows:

```
passthru: <language>.<suffix>
```

The password-protected folder example has two passthru sections. The first passthru is "C.h," which passes the code block to the C binding file *pwfolder.h*. This block of code defines a *DebugBox* macro, which can be used anywhere in the code for the new class.

```
#
# Passthru a debug message box to the .ih file
# (for inclusion in the .c file)
#
passthru: C.h, after;

#define DebugBox(Title, Text) WinMessageBox(HWND_DESKTOP,
                                            HWND_DESKTOP,
                                            (PSZ)Text,
                                            (PSZ)Title,
                                            0,
                                            MB_OK |
                                            MB_INFORMATION)

endpassthru;
```

The second passthru block is "C.ph"; this passes the code block to the C binding file *pwfolder.ph*. This block is used to define a data structure that is accessed by the private methods *_GetInfo* and *_SetInfo*, and is used to pass information to and from the dialog procedure that prompts the user for the folder password.

```
#
# Passthru private definitions to the .ph file
# (for inclusion in the .c file)
#
passthru: C.ph;

typedef struct _PWF_INFO {
                          CHAR   szPassword[20];
                          CHAR   szCurrentPassword[20];
                          CHAR   szUserid[20];
                          } PWF_INFO;
typedef PWF_INFO *PPWF_INFO;

endpassthru;
```

14.3.2.7 Data Section

This section lists the instance variables used by the class. In the password-protected folder example, three variables are defined as follows:

```
#
# Define instance data for the class
#
data:
CHAR szPassword[20];
--  This is the password that locks the folder
CHAR szCurrentPassword[20];
--  This is the password the user has entered to be
--  checked against the lock password
CHAR szUserid[20];
--  The userid data is here for future expansion
```

Note that the *szUserid* instance variable is not used in the version discussed in this document, since the current example assumes only a single workstation user. However, it is feasible for user identification to be obtained at startup, and held by the system for authentication against a password to determine whether access is permitted.

14.3.2.8 Methods Section

The last section in the class definition file contains a list of all the methods to be defined by the object class. ANSI C function-prototype syntax is used to define each method. When coding these definitions, it is recommended that the methods be divided into the following parts:

1. Methods that are new for this class
2. Methods that are overridden from ancestor classes

The following section shows two methods taken from the folder example's class definition file.

This first method will be used in the password dialog to take a copy of the object's instance data and place it in a structure that the dialog code may access.

```
#
# Define new methods
#
methods:

BOOL QueryInfo(PPWF_INFO pPWFolderInfo), private;
--
-- METHOD:   QueryInfo                                    PRIVATE
--
-- PURPOSE:  Copy the PWFolder instance data into
--           the PWF_INFO structure that pPWFolderInfo
--           points to.
--
```

The second example shows an overridden method. This method originates in the *WPObject* class, which is a base class. It is used to set up the password string when the folder object is created.

```
#
# Specify methods being overridden
#

override wpSetup;
--
-- OVERRIDE: wpSetup                                      PUBLIC
--
-- PURPOSE:  Here we can set the folder password
--           to that passed in from the object
--           create command.
--
```

More detailed information on class definition files and the OIDL is given in the *IBM SOM Programming Reference*.

14.3.3 C Implementation of an Object Class

When the SOM Precompiler has been run successfully against a class definition file, it will produce all the source files necessary to build a Workplace Shell DLL. The most important of these files for the C programmer is the C source code file, which has an extension of .C. This file contains definitions and "function stubs" for all the methods defined by the class. This file must be edited by the programmer to add the actual application logic to each method. Figure 157 on page 237 shows the SOM Precompiler-generated function stub for the *QueryInfo* method from the folder example.

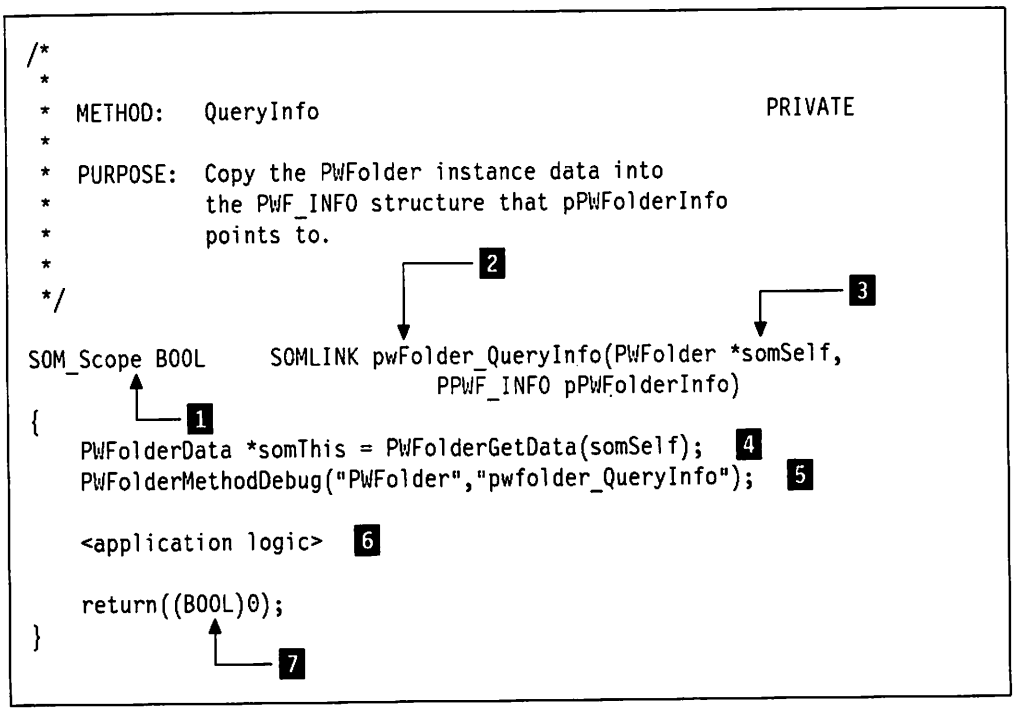


Figure 157. A SOM Precompiler-generated Function Stub

Notes:

1 SOM_Scope declares the function scope according to the language being used. For example, in C++, SOM_Scope would be defined as *extern C* but in C it is simply defined as *extern*.

2 It can be seen that the external prefix "pwfolder_," which was specified in the class definition file, has been placed in front of the function as expected. Note that the SOM Precompiler generates a macro for this function in the private header file:

```
#define _QueryInfo PWFolder_QueryInfo
```

This avoids the necessity for the programmer to type the full function name, and helps make the code more readable.

3 Since SOM uses the C language, methods from SOM objects cannot be referenced in a very elegant manner. The first parameter to a SOM method must be a pointer to an object that can invoke that method. In the actual method function, this pointer is given the name *somSelf*. For example, the difference between C and C++ is as follows:

```
/* Let us say */

     pMyObject =  (pointer to an object);

// in C++ the following syntax may be used

     pMyObject->Method(param1, param2....);

/* but in C the following is required */

     Method(pMyObject, param1, param2....);
```

4 This statement uses the pointer to the object to initialize a pointer to access the object's instance data. See 14.2.1.2, "Method Processing and Instance Data" on page 222 for further information on instance data.

5 This line will perform tracing. Tracing is switched on whenever the SOM global variable *SOM_TraceLevel* is set to a non-zero value.

6 This section is left blank by the SOM Precompiler for the developer to fill with the application logic. This logic may include access to system and/or Presentation Manager resources. For the password-protected folder example, the *_QueryInfo* method must copy the instance variables to the PWF_INFO data structure defined in the passthru section of the class definition file. The code required to do this is as follows:

```
strcpy(pPWFolderInfo->szPassword, _szPassword);
strcpy(pPWFolderInfo->szCurrentPassword, _szCurrentPassword);
strcpy(pPWFolderInfo->szUserid, _szUserid);
```

This code must be inserted in the C source file by the programmer, after the file is generated by the SOM Precompiler. This may be done using a normal text editor.

7 Finally, the SOM Precompiler provides a default zero return statement, typecast with the return data type of the method as declared in the methods section of the class definition file. This statement may be altered by the programmer if required, provided that consistency with the method's prototype and declaration is maintained.

14.4 Object Behavior

The behavior of an object in the Workplace Shell is very similar to that of a window under Presentation Manager. An object must have its class registered with the system, an instance of that class must be created ("instantiated") in the system, and that instance (and any other instance) then receives messages and uses its methods to process these messages. When processing is completed, the instance may be destroyed.

One significant difference between a Workplace Shell object class and a window class under Presentation Manager is that Workplace Shell object classes are

normally **persistent**; that is, while a Presentation Manager window class is defined only for the duration of the application's execution, a Workplace Shell object class remains defined to the system, and is usable by any application until such time as it is explicitly deregistered from the system.

14.4.1 Creating an Object

A new object class in the Workplace Shell is typically created by taking an existing object class and subclassing it, introducing new data and methods, and modifying existing behaviors where required. The new object class is then registered with the Workplace Shell, and is available from that point on.

14.4.1.1 Registration

Once an object class has been defined, compiled and placed into a dynamic link library, it must be registered with Workplace Shell before it can be used. This may be accomplished in any of two ways:

- An object class may be registered with the Workplace Shell using the **WinRegisterObjectClass()** function. This function records the name of the object class, and the name of the DLL that contains the code to implement the class. Note that if you are specifying a fully qualified path name for *pszModName*, then the DLL does not need to be placed in the LIBPATH.
- Additionally an object may also be registered with the Workplace Shell using the **SysRegisterObjectClass()** function from REXX. Like the WinRegisterObjectClass, this function also records the name of the object class, and the name of the DLL that contains the code to implement the class.

An example of the **WinRegisterObjectClass()** function is given in Figure 158, and an example of the **SysRegisterObjectClass()** function is given in Figure 159 on page 240.

```
PSZ  pszClassName = "NewObject";                /* Class name           */
PSZ  pszModName = "NEWOBJ";                     /* DLL module for class */
BOOL bSuccess;                                  /* Success flag         */

bSuccess = WinRegisterObjectClass(pszClassName, /* Register class       */
                                  pszModName);  /* DLL module name      */
```

Figure 158. Registering a Workplace Shell Object Class

Figure 158 provides a very simple example. A useful technique for registering object classes is to build a simple program that reads a set of strings from an ASCII data file and uses these strings as parameters to the **WinRegisterObjectClass()** function. In this way, a generic object-registration routine can be built and used for multiple object classes, without the need to modify and recompile source code.

Figure 159 on page 240 shows a sample piece of REXX code that registers a class called pwFolder to the Workplace Shell. Notice that the DLL which contains the pwFolder Workplace Object is also copied from the current directory. If this copy was unsuccessful, the author of this code assumed this was because the Workplace Shell has the DLL opened and so the REXX code deregisters the class from the Workplace Shell.

```
/*  */
Call RxFuncadd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
Call SysLoadFuncs

'@echo off'

'copy pwfolder.dll c:\os2\dll 1>nul: 2>nul:'

if rc then do
   say 'Error DLL could not be updated please re-boot'
   /* Remove bad entry */
   RetCode = SysDeregisterObjectClass( "PWFolder");
   'pause'
   exit(1)
   end

RetCode = SysRegisterObjectClass( "PWFolder", "pwfolder")

if RetCode then
   say 'PWFolder Class registered'
else do
   say 'Error PWFolder Class failed to register'
   /* Remove false entry */
   RetCode = SysDeregisterObjectClass( "PWFolder");
   exit(1)
   end
```

Figure 159. REXX Code to Register a Workplace Object

Note that once an object class has been registered with the Workplace Shell, it is permanently available until it is explicitly deleted by deregistering it. See 14.4.4, "Deregistering an Object Class" on page 264 for information on deregistering an object class.

14.4.1.2 Class Data

Class data is owned by the object class rather than by an instance of that class. It is therefore available to all instances of the class, and must be initialized prior to instantiating any objects within the class.

For this reason, class data is initialized when the object classes are loaded from their DLLs, either during Workplace Shell initialization or dynamically during execution. Class data initialization is performed by the _wpclsInitData class method, which is called by the system when the class is loaded. If a new object class has class data that must be initialized, it should override the _wpclsInitData method and perform its class-specific processing.

An example of an overridden _wpclsInitData method from the password-protected folder example is shown in Figure 160 on page 241.

```
HMODULE   hModule;
   :
   :
SOM_Scope void SOMLINK pwfoldercls_wpclsInitData(M_PWFolder *somSelf)
{
    CHAR  ErrorBuffer[100];                    /* Error buffer          */

    /* M_PWFolderData *somThis =
                      M_PWFolderGetData(somSelf); */
    M_PWFolderMethodDebug("M_PWFolder",        /* Set debug info        */
                          "pwfoldercls_wpclsInitData");

    DosLoadModule((PSZ) ErrorBuffer,           /* Get module handle     */
                  sizeof(ErrorBuffer),         /* Size of error buffer  */
                  "PWFOLDER",                  /* Name of DLL           */
                  &hModule);                   /* Module handle         */

    parent_wpclsInitData(somSelf);             /* Allow default proc    */
}
```

Figure 160. Initializing Class Data

In the example shown in Figure 160, a global variable *hModule* is used to contain the module handle for the DLL, which is required when loading Presentation Manager resources such as strings, pointers or dialogs. Since a global variable is used rather than a class data variable, the first statement in the overridden method, which obtains a handle to the class data, is not required and is therefore commented out.

Any class data items obtained or initialized by an object class from within the _wpclsInitData method should also be freed by the object class, by overriding the _wpclsUnInitData method. This method is invoked by the system when an object class is deregistered (see 14.4.4, "Deregistering an Object Class" on page 264), or when the Workplace Shell process is terminated. An example of the _wpclsUnInitData method is shown in Figure 161.

```
SOM_Scope void   SOMLINK pwfoldercls_wpclsUnInitData(M_PWFolder *somSelf)
{
    /* M_PWFolderData *somThis
                     = M_PWFolderGetData(somSelf); */
    M_PWFolderMethodDebug("M_PWFolder",        /* Set debug info        */
                          "pwfoldercls_wpclsUnInitData");

    DosFreeModule(hModule);                    /* Free module handle    */

    parent_wpclsUnInitData(somSelf);           /* Allow default proc    */
}
```

Figure 161. Freeing Class Data Items

The example shown in Figure 161 assumes that the module handle for the DLL has already been obtained and stored in the global variable *hModule*, as shown in Figure 160.

14.4.1.3 Instantiation

Once an object class has been registered with the Workplace Shell, an instance of that class may be created; this is known as **instantiation**. This may be done in one of three ways. One of the simplest method is to open the *Templates* folder and drag the template for the object class to the required location. Alternatively, an object may be created from within an application using the WinCreateObject() function. An example of this is shown in Figure 162. And lastly Figure 163 shows a sample piece of REXX code that creates a Workplace Object called pwFolder, along with some parameters for the object.

```
PSZ  pszClassName   = "NewObject";                 /* Class name           */
PSZ  pszObjectTitle = "My New Object";            /* Object title         */
PSZ  pszParams = "ICON=C:\\ICONS\\MYNEWICON.ICO"; /* Setup string         */
PSZ  pszLocation = "C:\\Desktop\\MyNewFolder";    /* Location for object  */

ULONG ulFlags;                                    /* Creation flags       */

HOBJECT hObject;                                  /* Object handle        */

hObject = WinCreateObject(pszClassName,           /* Create object        */
                          pszObjTitle,            /* Title for icon       */
                          pszParams,              /* Setup string         */
                          pszLocation,            /* Location for object  */
                          CO_REPLACEIFEXISTS);    /* Creation flags       */
```

Figure 162. "C" Code to Create an Object

```
/*  */
Call RxFuncadd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
Call SysLoadFuncs

'@echo off'

RetCode = SysCreateObject( "PWFolder", "FinanceFile", "<WP_DESKTOP>",
                           "PASSWORD=wps;OBJECTID=<MyFinanceFile>")

if RetCode then
   say 'PWFolder Object created'
else do
   say 'Error creating object'
   exit(1)
   end
```

Figure 163. REXX Code to Create an Object

Note that the *pszParams* parameter shown in Figure 162 is used to contain a *setup string*, which can be used to pass one or more of a number of parameters to the object class. In the example, it is used only to set the icon for the object, but may also be used to specify other parameters for that instance of the class. The keywords and values supported by the *WPObject* class are documented in the *IBM OS/2 Version 2.0 Presentation Manager Reference*; other object classes may add their own keywords and values.

The final parameter contains one or more flags which determine the behavior of the WinCreateObject() call if the object being created clashes with an object that

already exists with the specified name and in the specified location. Valid actions are for the call to fail, to update the existing object or to replace the existing object. These flags are documented in the *IBM OS/2 Version 2.0 Presentation Manager Reference*.

The setup string is passed as a parameter to the method, which can either be invoked when the object is instantiated, or during a call to WinSetObjectData. Because a call can be made to the _wpSetup method from WinSetObjectData, you must not process any default settings other than those related to the parameters passed to the _wpSetup method. This method is defined by the *WPObject* class, and may be overridden by a new object class in order to check for its own keywords and take appropriate setup action.

The _wpSetup method accepts the setup string as a parameter, and may then parse the setup string, extract any class-specific data and perform appropriate processing on that data. However, since many of the keywords that may be specified in the setup string are defined by the *WPObject* class and are handled by the default _wpSetup method, the default processing must be carried out. In this particular case, the default processing may be carried out before or after the class-specific processing.

An example of an overridden _wpSetup method is shown in Figure 164 on page 244; this example shows the use of an additional parameter in the setup string (PASSWORD=) to set an initial password for a password-protected folder upon folder creation. The setup string is parsed from within the object by calling the _wpScanSetupString method. Both of these methods, along with the keywords supported by the *WPObject* class, are described in the *IBM OS/2 Version 2.0 Presentation Manager Reference.*

After performing the class-specific processing in the _wpSetup method, an object class should invoke its parent class's _wpSetup method to perform the default processing for any other keywords in the setup string that are defined by the parent class.

Before the _wpSetup method is invoked, the system invokes the object's _wpInitData method, which allows an object to allocate resources and initialize its instance data. See 14.4.1.4, "Instance Data" on page 244 for further details.

Note that unlike a Presentation Manager window, which exists only for the duration of an application's execution, an object remains in existence permanently unless explicitly deleted from the system.

```
SOM_Scope BOOL SOMLINK pwfolder_wpSetup(PWFolder *somSelf,
                                        PSZ pszSetupString)
{
    CHAR  pszInitPword[20];                    /* Buffer for password   */
    BOOL  bFound;                              /* Success flag          */
    ULONG ulRetLength;

    PWFolderData *somThis =                    /* Get instance data     */
                PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",            /* Set debug info        */
                        "pwfolder_wpSetup");

    if (pszSetupString != NULL)                /* If string is present  */
       {
       bFound=_wpScanSetupString(somSelf,       /* Scan setup string to */
                                 pszSetupString, /* find keyword        */
                                 "PASSWORD"
                                 pszInitPword,
                                 &RetLength);

       if (bFound)                             /* If parameter present  */
          {
          strcpy(_szPassword,                  /* Copy p'word to folder */
                 pszInitPword);                /* p'word and current    */
          strcpy(_szCurrentPassword,           /* p'word - initialize   */
                 pszInitPword);                /* in unlocked state     */
          }
       }
    return (parent_wpSetup(somSelf,            /* Allow default proc to */
                           pszSetupString));   /* occur                 */
}
```

Figure 164. Object Setup. This example shows an overridden _wpSetup method which parses the setup string to extract class-specific parameters.

14.4.1.4 Instance Data

When an object is created or awakened from a dormant state, the _wpInitData method is invoked by the system. This method allows an object to initialize its instance data to a known state. Operating system resources should be allocated at this stage, but Presentation Manager resources should not, since a view of the object is not yet being opened. The allocation of Presentation Manager resources is typically done during processing of the _wpOpen method (see 14.4.2.1, "Opening an Object" on page 245).

If an object has its own instance data, which must be initialized to a known state before processing may be carried out, the object should override the _wpInitData method in its class definition file, and include the initialization code. However, for any object class other than a base storage class, the default initialization processing must be carried out in addition to the class-specific processing. This allows the correct initialization of any instance data items defined by the parent class, and ensures that the new object class behaves in a manner consistent with its ancestors.

Figure 165 on page 245 shows an overridden _wpInitData method, which initializes the password information for a password-protected folder.

```
SOM_Scope void SOMLINK pwfolder_wpInitData(PWFolder *somSelf)
{
    CHAR  ErrorBuffer[100];                     /* Error data buffer     */

    PWFolderData *somThis =                     /* Get instance data     */
                PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",             /* Set debug info        */
                        "pwfolder_wpInitData");

    strcpy(_szCurrentPassword,                  /* Initialize folder     */
           "password");                         /* password              */
    strcpy(_szPassword,                         /* Set current password  */
           "password");                         /* to folder password    */
                                                /* ie. Set unlocked      */

    return(parent_wpInitData(somSelf));         /* Perform default proc  */
}
```

Figure 165. Initializing Instance Data

Note that during processing of the _wpInitData method, the instance data of the object is not necessarily in a known state. The programmer must therefore take great care when carrying out any processing during the execution of this method, in order to avoid using data that may not yet have been initialized correctly. Failure to follow this guideline may cause unpredictable results for the object.

14.4.2 Using an Object

A user typically accesses an object by opening a "view" of that object. For example, to access the contents of a folder object, the user opens the default view (usually an icon view) of the folder, which then displays its contents. This is certainly true for container objects such as folders, and for the password-protected folder class used as an example in this chapter, although other "device" objects such as printers or the shredder may be used without a view.

When no view of an object is open, *and* the folder within which the object resides is not open, the object is said to be "dormant"; typically, no system resources are allocated to the object and its instance data is in an unknown state. Opening and closing views of an object therefore involve not only the opening and closing of windows, but also allocating and freeing resources, and saving and restoring the instance data of the object. Similarly, opening a folder requires saving and restoring the instance data of the objects in that folder.

14.4.2.1 Opening an Object

As mentioned above, a user typically interacts with an object using a view of that object. An object may support various types of view; for example, the *WPFolder* object class supports icon, tree, details and settings views. By default, an object class supports the view types defined by its ancestors, and a programmer may also define new view types for the object class.

When a view of an object is opened, the _wpViewObject method is invoked by the Workplace Shell. This method determines if there is already an open view of the view specified for the object. If there is not then _wpOpen is called to open a view for the object. If there already is an open view, _wpViewObject checks the

objects settings for concurrent views. If concurrent views are set, the _wpOpen is called to open an additional view of the object with the specified view. Therefore your programs should call _wpViewObject and not _wpOpen, but override _wpOpen to add your own unique views. Note that the concurrent view setting can normally be found on an object's Settings notebook, under the "Window" tab, and is labelled "Object Open Behavior".

The _wpOpen method is defined and implemented by the base storage class *WPObject*, and may be overridden by a new object class to perform its own class-specific processing. The supported views for each object class are implemented as part of the _wpOpen method, using Presentation Manager windows.

When a view is opened by the user from a context menu, the _wpMenuItemSelected method is invoked (see 14.2.1.6, "Attaching a Method to the Context Menu" on page 224 for more detailed discussion of this method). The _wpMenuItemSelected method typically invokes the _wpViewObject method, which may invoke the _wpOpen method as outlined above.

When the user opens a view by double-clicking the mouse on an object's icon, the _wpViewObject method invokes the _wpOpen method and passes an OPEN_DEFAULT value. The default processing for the _wpOpen method invokes the _wpQueryDefaultView method to determine the default view for the object, and immediately invokes the _wpOpen method a second time with the identifier for that view.

An example of an overridden _wpOpen method is given in Figure 166 on page 247. This example shows a password-protection facility being added to a folder to prevent access by unauthorized users. Upon invocation of the _wpOpen method, the password-protected folder object class displays a dialog box to accept a password from the user. It then compares that password with the correct password for that folder before actually opening the folder. Visual cues such as the folder's icon and the word "Locked" on the folder's title are modified or removed during the _wpOpen processing.

```
SOM_Scope HWND SOMLINK pwfolder_wpOpen(PWFolder *somSelf,
                                       HWND hwndCnr,
                                       ULONG ulView,
                                       ULONG param)
{   ULONG ulResult;
    CHAR  szTitle[100];
    PDLGDATA pDlgData;
    PWFolderData *somThis =
                 PWFolderGetData(somSelf);      /* Set instance data     */
    PWFolderMethodDebug("PWFolder",             /* Set debug info        */
                        "pwfolder_wpOpen");
    if ((strcmp(_szCurrentPassword,             /* If not locked         */
                _szPassword)) == 0)
       return(parent_wpOpen(somSelf,            /* Allow open to proceed */
                            hwndCnr,            /* in normal way, using  */
                            ulView,             /* default processing    */
                            param));

    /* Allocate memory to be passed to the password dialog */
    pDlgData = (PDLGDATA)_wpAllocMem(somSelf, sizeof(DLGDATA), NULL);
    pDlgData->cbSize = sizeof(DLGDATA);
    pDlgData->somSelf = somSelf;
    ulResult = WinDlgBox(HWND_DESKTOP,          /* Display p'word dialog */
                         HWND_DESKTOP,          /* Desktop is owner      */
                         dpPassword,            /* Dialog procedure      */
                         hModule,               /* Module handle         */
                         DLG_PASSWORD,          /* Dialog resource id    */
                         pDlgData);            /* Object pointer        */
    if (ulResult == DID_OK)                     /* If not cancelled      */
       {
       if ((strcmp(_szCurrentPassword,          /* If correct password   */
                   _szPassword)) == 0)
          {
          strcpy(szTitle,                       /* Get title string      */
                 _wpQueryTitle(somSelf));
          szTitle[strlen(szTitle)-9] = '\0';    /* Remove <LOCKED>       */
          _wpSetTitle(somSelf,szTitle);         /* Reset title string    */
          return (parent_wpOpen(somSelf,        /* Allow default _wpOpen */
                                hwndCnr,        /* processing to occur   */
                                ulView,         /* by invoking parent's  */
                                param));        /* method                */
          }
       else
          {
          WinMessageBox(HWND_DESKTOP,           /* Display message box   */
             HWND_DESKTOP,
             "Password incorrect. Folder remains locked.",
             "Password Failed",
             0, MB_OK | MB_CUAWARNING);
          return((BOOL)0);                      /* Return FALSE          */
          }
       }
}
```

Figure 166. Opening an Object. This example shows the _wpOpen method, which is called by the system when a view of an object is opened, being overridden to add password protection to a folder.

Since the view being opened in this case is a view defined by the *WPFolder* class, the actual opening of the view and presentation of the folder's contents is handled using the default processing supplied by the parent class, which is called after the class-specific processing has completed.

If an object class wishes to create a new view, it must add the name of the view to the *Open* submenu in the object's context menu, and include a case for that view in the _wpMenuItemSelected method. This method then invokes _wpViewObject with a specific value in the *ulView* parameter, indicating the view to be opened. The class-specific processing for _wpOpen must test for this value, open a window and display the correct information using Presentation Manager functions.

The example in Figure 166 does not include the code to set the folder's icon to the "unlocked" state. This code is identical to the code used in Figure 151 on page 224 to set the icon to the "locked" state; the resource identifier of the "unlocked" icon is simply substituted in the _wpOpen method for the identifier of the "locked" icon.

Note that in many cases, it is important for an object class to allow the default processing for _wpOpen to occur *before* it attempts to carry out its own processing. This allows instance data and control information to be established and initialized before the object attempts any processing using these items. In Figure 166 on page 247 however, the additional class-specific processing determines whether the object should open *at all*; if processing is allowed to proceed, no alteration to the default processing takes place. The default processing may therefore be carried out after the additional class-specific processing introduced by the password-protected folder class.

The default processing for the _wpOpen method supports a number of views, depending upon the parent class of the object; for example, the processing for the *WPFolder* class supports ICON, TREE and DETAILS views. For new object classes which support additional views, the _wpOpen method must be overridden and the additional view types opened explicitly as windows using appropriate Presentation Manager functions. Since a view of an object is essentially a window, new views can be implemented as normal Presentation Manager windows and the correct information displayed using text or graphical programming functions, according to the requirements of the object class.

The application must always define a new view if it introduces one. Never process OPEN_DEFAULT other than passing it to the parent class. If you want to have your own view be the default, then override the _wpclsQueryDefaultView method.

Note that upon opening a view using a Presentation Manager window, an object should add itself to the "Use List" maintained by the Workplace Shell. If the view is the first view of the object to be opened, this causes the Workplace Shell to modify the object's icon to indicate the "in use" state. The object should also register the view with the Workplace Shell, which will then subclass the view's frame window, automatically attach the object's context menu to the window's system menu icon, and add the view to the Workplace Shell's Window List. These steps are done using the _wpAddToObjUseList and _wpRegisterView methods, as shown in Figure 167 on page 249.

```
HWND    hView;                                  /* View window handle    */

typedef struct _OBJECTVIEW                      /* Object view structure */
               {
               SOMAny   *Object;                /* Object pointer        */
               USEITEM  UseItem;                /* USEITEM structure     */
               VIEWITEM ViewItem;               /* VIEWITEM structure    */
               } OBJECTVIEW;

OBJECTVIEW *pObjectView;                        /* Pointer to structure  */

<Create Window>                                 /* Get window handle     */

pObjectView = _wpAllocMem(somSelf,              /* Allocate memory       */
                          sizeof(OBJECTVIEW),   /* Size of mem object    */
                          NULL);
pObjectView->Record          = somSelf;         /* Initialize OBJECTVIEW */
pObjectView->UseItem.type    = USAGE_OPENVIEW;  /* structure             */
pObjectView->ViewItem.view   = OPEN_CUST;
pObjectView->ViewItem.handle = hView;

WinSetWindowULong(hView,                        /* Store pointer to      */
                  QWL_USER,                     /* structure in window   */
                  (ULONG)pObjectView);          /* words                 */

_wpAddToObjUseList(somSelf,                     /* Add to Use List       */
                   &pObjectView->UseItem);      /* USEITEM structure     */
_wpRegisterView(somSelf,                        /* Register view         */
                hView,                          /* View window handle    */
                "Customer Details")             /* Title of view         */
```

Figure 167. Opening a Custom View of an Object

The Workplace Shell makes use of a USEITEM and a VIEWITEM structure in the _wpAddToObjUseList method. It assumes that these structures are contiguous in memory; hence they should be allocated as part of a larger data structure such as the OBJECTVIEW structure shown in Figure 167. A pointer to this structure is stored in the window words of the view window, so that information such as the object's pointer can be accessed from the view's window procedure.

Note that upon closing a view, the view's window procedure should invoke the _wpDeleteFromObjUseList method to remove the view from the Use List. If the view is the only open view of the object, the object's icon is modified to remove the "in use" emphasis.

14.4.2.2 A Custom View of our pwFinanceFile

This section shows how to implement a custom view of the pwFinanceFile Workplace Object.

Figure 168 on page 250 shows how the "Open Finance File" menu item is added onto the "Open" menu item, which appears on the pwFinanceFile's context menu. Note that we also add the "Lock Finance File" menu item. Figure 169 on page 251 shows the resulting context menu which is provided by Figure 168 on page 250. Note that the "OS/2 System Editor" menu item is also present. The Workplace Shell wpDataFile class has added this due to file type associations. In this case because this object does not have a file type, it is assumed by the

ancestor classes to be text and the default association for the text type is the OS/2 System Editor. We could remove this by overriding the _wpclsQueryDefaultView method, thus making our view the default one.

```
/*
 *
 *   METHOD:   wpModifyPopupMenu                              PUBLIC
 *
 *   PURPOSE:  Adds an additional "Lock" item to the object's context menu.
 *             Adds a "Open Finance File" item to the "Open" item
 *   INVOKED:  By Workplace Shell, upon instantiation of the object instance.
 *
 */

SOM_Scope BOOL
          SOMLINK pwFinanceFile_wpModifyPopupMenu(PWFinanceFile *somSelf,
                HWND hwndMenu,
                HWND hwndCnr,
                ULONG iPosition)
{
    PWFinanceFileData *somThis =                    /* Get instance data pointer */
                 PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile",             /* Set debug info  */
                        "pwFinanceFile_wpModifyPopupMenu");

    _wpInsertPopupMenuItems(somSelf,            /* Insert menu item          */
                            hwndMenu,           /* Menu handle               */
                            iPosition,          /* Default position          */
                            hmodThisClass,      /* Module handle             */
                            ID_CXTMENU_LOCK,    /* Menu item identifier      */
                            0);                 /* No submenu identifier     */

    _wpInsertPopupMenuItems( somSelf,           /* Insert menu item          */
                             hwndMenu,          /* Menu handle               */
                             0,                 /* at the top!               */
                             hmodThisClass,     /* Module handle             */
                             ID_OPENFinanceFile, /* Menu item identifier     */
                             WPMENUID_OPEN);    /* Submenu identifier        */


    return(parent_wpModifyPopupMenu(somSelf,    /* Invoke default processing */
                                    hwndMenu,
                                    hwndCnr,
                                    iPosition));
}
```

Figure 168. _wpModifyPopupMenu .C Code. This example shows how to add two menu items to the "Open Finance File" menu item on the pwFinanceFile's context menu.

Figure 171 on page 252, Figure 172 on page 253, Figure 173 on page 255 and Figure 174 on page 258 show the additional code required to process the selection of the "Open Finance File" menu item. Figure 170 on page 251 shows the window view which is presented when the user selects the "Open Finance File" menu item. This window is empty and the population of the window with information from the file associated with the instance of the pwFinanceFile is a normal PM programming exercise which is left for the reader to perform.

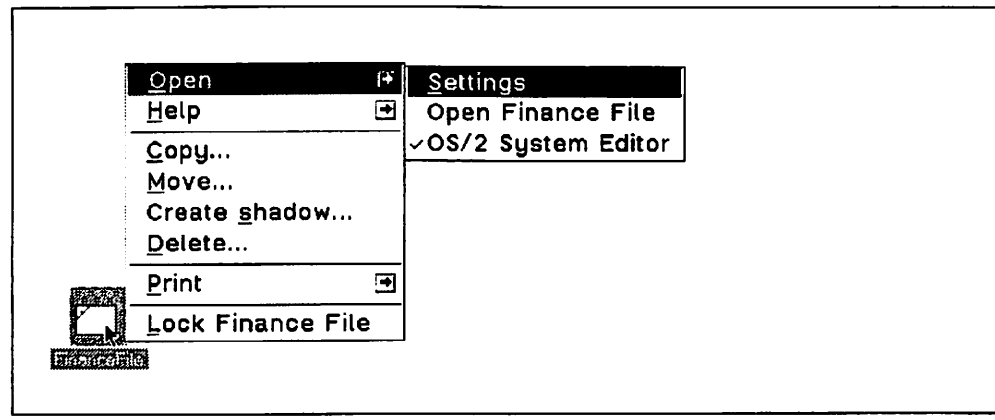


Figure 169. pwFinanceFile's Context Menu

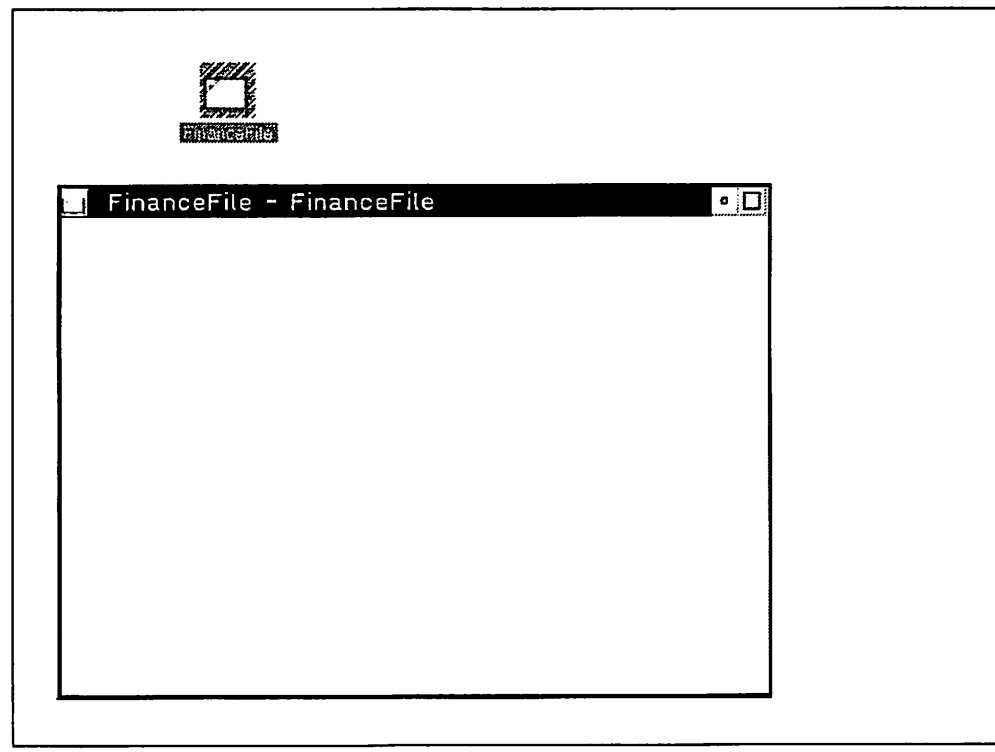


Figure 170. pwFinanceFile's Custom View

```
/*
 *
 *   METHOD:   wpMenuItemSelected                          PUBLIC
 *
 *   PURPOSE:  Processes the user's selections from the context menu.  The
 *             overridden method processes the added "Lock" & "OPENFinanceFile"
 *             items, and passes all others to the parent method
 *
 *   INVOKED:  By Workplace Shell, upon selection of a menu item by the user.
 *
 */

SOM_Scope BOOL
          SOMLINK pwFinanceFile_wpMenuItemSelected(PWFinanceFile *somSelf,
                HWND hwndFrame,
                ULONG ulMenuId)
{
    PWFinanceFileData *somThis =           /* Get instance data pointer */
                 PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile",         /* Set debug info*/
                        "pwFinanceFile_wpMenuItemSelected");

    switch( ulMenuId )                              /* Switch on item identifier */
    {
       case IDM_LOCK:                               /* Lock item selected  */
          _LockFinanceFile(somSelf);      /* Invoke _LockFinanceFile method */
          break;

    /*
     *   We could call wpOpen here, but if the object is already opened
     *   the following API determines whether the object should be
     *   resurfaced or if multiple views are desired.
     *   Must call wpViewObject not wpOpen.  If you use wpOpen then multiple
     *   concurrent views won't work.  User can set object to open multiple
     *   views or switch to.
     */

       case IDM_OPENFinanceFile:                    /* Open a view selected      */
          _wpViewObject(somSelf, NULLHANDLE, OPEN_FinanceFile, 0);
          break;


       default:                                     /* All other items           */
          parent_wpMenuItemSelected(somSelf,   /* Invoke default processing */
                                    hwndFrame,
                                    ulMenuId);
          break;
    }
}
```

Figure 171. _wpMenuItemSelected .C Code

"C" Code

```
/*
 *
 *   METHOD:   wpOpen                                        PUBLIC
 *
 *   PURPOSE:  Only allows a FinanceFile to be opened if the FinanceFile is unlocked, or
 *             if the user supplies the correct password in response to the
 *             dialog.
 *
 *   INVOKED:  By Workplace Shell, upon selection of the "Open" menu item by
 *             the user.
 *
 */

SOM_Scope HWND   SOMLINK pwFinanceFile_wpOpen(PWFinanceFile *somSelf,
                HWND hwndCnr,
                ULONG ulView,
                ULONG param)
{
    ULONG    ulResult;                          /* Return value             */
    CHAR     szTitle[100];             /* FinanceFile title buffer       */
    PVOID    pCreateParam;
    BOOL     bAllowAccess = FALSE;              /* user is allowed in */
    PDLGDATA pDlgData;                          /* pCreateParam structure */

    PWFinanceFileData *somThis =                     /* Get instance data pointer */
                PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile",             /* Set debug info           */
                        "pwFinanceFile_wpOpen");

    if ((strcmp(_szCurrentPassword,             /* If FinanceFile is locked   */
         _szPassword)) != 0)
    {
       somPrintf("ask for a password\n");

    /* Allocate memory to be passed to the password dialog */
    pDlgData = (PDLGDATA)_wpAllocMem(somSelf, sizeof(DLGDATA), NULL);
    pDlgData->cbSize = sizeof(DLGDATA);
    pDlgData->somSelf = somSelf;

    ulResult = WinDlgBox(HWND_DESKTOP,          /* Display password dialog   */
                         HWND_DESKTOP,          /* Desktop is owner          */
                         PasswordDlgProc,       /* Dialog procedure address  */
                         hmodThisClass,         /* Module handle             */
                         ID_DLG_PASSWORD,       /* Dialog resource id        */
                         pDlgData);             /* Create Param holding      */
                                                /* pointer to this object    */
```

Figure 172 (Part 1 of 2). _wpOpen

```
    if (ulResult == DID_OK )                        /* If user hit OK button     */
       {
       if ((strcmp(_szCurrentPassword,              /* If password is correct    */
                _szPassword)) == 0)
          {
          strcpy(szTitle,                           /* Get title string          */
                 _wpQueryTitle(somSelf));
          szTitle[strlen(szTitle)-9] = '\0'; /* Remove <LOCKED>     */
          _wpSetTitle(somSelf,szTitle);             /* Reset title string        */

          _wpSetIcon(somSelf,                       /* Set icon to unlocked      */
                     hUnlockedIcon);                /* state                     */

          /* now we can allow the user access to the object proper ! */
          bAllowAccess = TRUE;

          }
       else                                         /* Password is incorrect     */
          {
          WinMessageBox(HWND_DESKTOP,               /* Display message to user   */
                        HWND_DESKTOP,
                        "Password incorrect. FinanceFile remains locked.",
                        "Password Failed",
                        0,
                        MB_OK |
                        MB_CUAWARNING );
          return((HWND)0);                          /* Return NULL handle        */
          }
       }
    } else {
      bAllowAccess = TRUE;
    }
    if (bAllowAccess) {
       switch (ulView) {
       case OPEN_FinanceFile:
         if (!_wpSwitchTo(somSelf, ulView)) {
           /* Create a basic frame and client window for this instance */
           return PWFinanceFileInit(somSelf);
         } /* endif */
         break;

       default:
         return(parent_wpOpen(somSelf,              /* Allow open to proceed in  */
                              hwndCnr,              /* normal way using default  */
                              ulView,               /* processing                */
                              param));
       } /* endswitch */
    } else {
    } /* endif */
}
```

Figure 172 (Part 2 of 2). _wpOpen

```
/*****************************************************************************
*                                                                            *
*       ROUTINE:    PWFinanceFileInit()                                      *
*                                                                            *
*       DESCRIPTION:  PWFinanceFile Inisialisation                           *
*                                                                            *
*       RETURNS:    Handle of PWFinanceFile frame window, NULL if error      *
*                                                                            *
*****************************************************************************/
HWND PWFinanceFileInit (PWFinanceFile* somSelf)
{

   HAB  hab;                                          /* PM anchor block handle */
   HWND hwndFrame = NULLHANDLE;                          /* Frame window handle */
   HWND hwndClient = NULLHANDLE;
   PWINDOWDATA pWindowData;
   BOOL fSuccess;
   SWCNTRL    swcEntry;                                        /* Switch Entry */
   FRAMECDATA flFrameCtlData;                                 /* Frame Ctl Data */

   somPrintf("PWFinanceFileInit\n");

   hab = WinQueryAnchorBlock(HWND_DESKTOP);
   if (!WinRegisterClass( hab , szFinanceFileWindowClass, (PFNWP)FinanceFileWndProc ,
                          CS_SIZEREDRAW | CS_SYNCPAINT, sizeof(pWindowData)))
   {
      somPrintf("FinanceFileInit Failure in WinRegisterClass\n");
      return NULLHANDLE ;
   }

   /*
    *   Allocate some instance specific data in Window words of Frame window.
    *   This will ensure our window procedure can use this object's methods
    *   (our window proc isn't passed a * somSelf pointer).
    */
   pWindowData = (PWINDOWDATA) _wpAllocMem(somSelf, sizeof(*pWindowData), NULL);

   if (!pWindowData)
   {
      somPrintf("FinanceFileInit wpAllocMem failed to allocate pWindowData\n");
      return NULLHANDLE;
   }

   memset((PVOID) pWindowData, 0, sizeof(*pWindowData));
   pWindowData->cb = sizeof(*pWindowData);            /* first field = size */
   pWindowData->somSelf = somSelf;

   /* Create a frame window
    */
   flFrameCtlData.cb           = sizeof( flFrameCtlData );
   flFrameCtlData.flCreateFlags = FCF_SIZEBORDER | FCF_TITLEBAR | FCF_SYSMENU |
                                  FCF_MINMAX ;
   flFrameCtlData.hmodResources = hmodThisClass;
   flFrameCtlData.idResources   = ID_UNLOCK;
```

Figure 173 (Part 1 of 3). pwFinanceFile's Initialization Function

```
hwndFrame =                                          /* create frame window */
WinCreateWindow(
   HWND_DESKTOP,              /* parent-window handle                    */
   WC_FRAME,                  /* pointer to registered class name        */
   _wpQueryTitle(somSelf),    /* pointer to window text                  */
   0,                         /* window style                            */
   0, 0, 0, 0,                /* position of window                      */
   NULLHANDLE,                /* owner-window handle                     */
   HWND_TOP,                  /* handle to sibling window                */
   (USHORT) ID_FRAME,         /* window identifier                       */
   (PVOID) &flFrameCtlData,   /* pointer to buffer                       */
   NULL);      ;              /* pointer to structure with pres. params. */

if (!hwndFrame)
{
   somPrintf("FinanceFileInit Failure in WinCreateWindow\n");
   return NULLHANDLE;
}
hwndClient =          /* use WinCreateWindow so we can pass pres params */
WinCreateWindow(
   hwndFrame,                 /* parent-window handle                    */
   szFinanceFileWindowClass, /* pointer to registered class name        */
   NULL,                      /* pointer to window text                  */
   0,                         /* window style                            */
   0, 0, 0, 0,                /* position of window                      */
   hwndFrame,                 /* owner-window handle                     */
   HWND_TOP,                  /* handle to sibling window                */
   (USHORT)FID_CLIENT,        /* window identifier                       */
   pWindowData,               /* pointer to buffer                       */
   NULL);                     /* pointer to structure with pres. params. */

if (!hwndClient)
{
   WinDestroyWindow(hwndFrame);
   return NULLHANDLE;
}

WinSendMsg(hwndFrame,WM_SETICON,MPFROMP(_wpQueryIcon(somSelf)),NULL);
WinSetWindowText(WinWindowFromID(hwndFrame,(USHORT)FID_TITLEBAR),
                                                    _wpQueryTitle(somSelf));

/*
 * Restore the Window Position
 */
fSuccess =
WinRestoreWindowPos(
   szFinanceFileClassTitle,                                  /* class title */
   _wpQueryTitle(somSelf),                              /* object title */
   hwndFrame);
```

Figure 173 (Part 2 of 3). pwFinanceFile's Initialization Function

```
   if (!fSuccess)
   {
    SWP        swp;

      /* Get the dimensions and the shell's suggested
       * location for the window
       */
      WinQueryTaskSizePos(hab,0,&swp);

      /* Set the frame window position
       */
      swp.fl               = SWP_SIZE|SWP_MOVE|SWP_RESTORE|SWP_ZORDER;
      WinSetWindowPos(hwndFrame, HWND_TOP, swp.x, swp.y, swp.cx,
                      swp.cy, swp.fl);
   }

   WinShowWindow(hwndFrame,TRUE);

   return hwndFrame;                                          /* success */

}   /* end FinanceFileInit() */
```

Figure 173 (Part 3 of 3). pwFinanceFile's Initialization Function

```
/*****************************************************************************
*
*   FinanceFileWndProc()
*
*   DESCRIPTION: FinanceFile Window Procedure
*
*****************************************************************************/
MRESULT EXPENTRY FinanceFileWndProc( HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2 )
{
  ULONG         MenuId;
  PWINDOWDATA   pWindowData;
  HWND          hwndFrame;
  CHAR          acBuffer[10];
  BOOL          fSuccess;
  CHAR          szPath[CCHMAXPATH];                                  //jt 55384
  ULONG         cbPath = CCHMAXPATH;                                        //jt 55384


   hwndFrame = WinQueryWindow(hwnd, QW_PARENT);

   switch( msg )
   {
      case WM_CREATE:

        pWindowData = (PWINDOWDATA) mp1;

        if (pWindowData == NULL)
        {
           somPrintf("FinanceFileWndProc:WM_CREATE couldn't get window words");
           return FALSE;
        }
        /*
         *   Fill in the class view/usage details and window specific data
         *   for this instance.
         */

        /*
         * Must create UseItem, add it to the object's use list and register the view
         */
        pWindowData->UseItem.type    = USAGE_OPENVIEW;
        pWindowData->ViewItem.view   = OPEN_FinanceFile;
        /*
         * Must be frame.  Be careful because this procedure is for the client.
         * Must get parent and pass that as ViewItem handle.
         */
        pWindowData->ViewItem.handle = hwndFrame;
        pWindowData->x               = 10;
        pWindowData->y               = 10;
        pWindowData->xDir            = 0;
        pWindowData->yDir            = 0;

        /*
         *   Set window pointer with object pointer and instance view info.
         *   Then add view to the in-use list so wpSwitchTo works.
         */
        WinSetWindowPtr(hwnd, QWL_USER, pWindowData);
```

Figure 174 (Part 1 of 3). pwFinanceFile's Window Procedure, FinanceFileProc()

```
        /*
         * _wpAddToObjUseList will tell the shell to store the view in
         * the internal linked list for the object to enable wpSwitchTo and other
         * methods to find the view.  The shell will also subclass the view window
         * this gives you title bar context menu when you call wpRegisterView.
         * wpRegisterView also puts the view in the window list and sets up
         * the title bar like:  "Object Title - View Title"
         */

        _wpAddToObjUseList(pWindowData->somSelf,&pWindowData->UseItem);
        _wpRegisterView(pWindowData->somSelf, hwndFrame,
                                              _wpQueryTitle(pWindowData->somSelf));
        WinSetFocus( HWND_DESKTOP, hwndFrame);

        /* what is the filename of the file */
        if (_wpQueryRealName(pWindowData->somSelf,szPath,&cbPath,TRUE))
           {
              somPrintf("File name is %s, size %i \n",szPath, cbPath);
           } else {
              somPrintf("Failed to get filename\n");
           } /* endif */

        break;

      case WM_COMMAND:

        break;

      case WM_PAINT:
         pWindowData = (PWINDOWDATA) WinQueryWindowPtr(hwnd, QWL_USER);

         if (pWindowData == NULL)
         {
            somPrintf("FinanceFileWndProc:WM_PAINT couldn't get window words\n");
            return FALSE;
         }
         else
         {
          HPS    hps;
          RECTL  rectl;

           hps = WinBeginPaint( hwnd, (HPS)NULLHANDLE, &rectl);
           WinFillRect( hps, &rectl, SYSCLR_WINDOW);
           WinEndPaint( hps );
         }
         break;
```

Figure 174 (Part 2 of 3). pwFinanceFile's Window Procedure, FinanceFileProc()

```
      case WM_CLOSE:
         {
          HAB hab;

            hab = WinQueryAnchorBlock(HWND_DESKTOP);

            pWindowData = (PWINDOWDATA) WinQueryWindowPtr(hwnd, QWL_USER);

            if (pWindowData == NULL)
            {
               somPrintf("FinanceFileWndProc:WM_CLOSE couldn't get window words\n");
               return FALSE;
            }
            fSuccess =
            WinStoreWindowPos(szFinanceFileClassTitle,_wpQueryTitle(pWindowData->somSelf),
                                                                    hwndFrame);
            /*
             * Must remove from the object UseList when window is closed.  (can be done
             * on WM_DESTROY instead)
             */
            _wpDeleteFromObjUseList(pWindowData->somSelf,&pWindowData->UseItem);
            _wpFreeMem(pWindowData->somSelf,(PBYTE)pWindowData);

            WinDestroyWindow ( hwndFrame ) ;
         }
         break;

      default:
         return WinDefWindowProc( hwnd, msg, mp1, mp2 );
   }
   return FALSE;

}  /* end FinanceFileWndProc() */
```

Figure 174 (Part 3 of 3). pwFinanceFile's Window Procedure, FinanceFileProc()

14.4.2.3 Automatic Opening Upon Instantiation

In many cases, it is desirable to automatically open a view of an object when the object is created. This may be achieved by using the OPEN= keyword in the setup string passed to the WinCreateObject() function. An example of this technique is shown in Figure 175.

```
PSZ  pszClassName   = "NewObject";              /* Class name           */
PSZ  pszObjectTitle = "My New Object";         /* Object title         */
PSZ  pszParams = "OPEN=ICON";                  /* Setup string         */
PSZ  pszLocation = "C:\\Desktop\\MyNewFolder"; /* Location for object  */

ULONG ulFlags = CO_UPDATEIFEXISTS;             /* Creation flags       */

HOBJECT hObject;                               /* Object handle        */

hObject = WinCreateObject(pszClassName,        /* Create object        */
                          pszObjTitle,         /* Title for icon       */
                          pszParams,           /* Setup string         */
                          pszLocation,         /* Location for object  */
                          ulFlags);            /* Creation flags       */
```

Figure 175. Automatically Instantiating an Object. This example shows the use of the OPEN= keyword to automatically open a view of an object upon creating the object.

The opening of the view specified in the OPEN= keyword is handled by the default processing for the _wpSetup method, as defined by the *WPObject* class. The default processing supports the icon, tree and details views, specified using

the ICON, TREE and DETAILS values for the OPEN= keyword respectively. For new object classes that support additional views, the _wpSetup method must be overridden and the additional view types opened explicitly as windows using appropriate Presentation Manager functions.

14.4.2.4 Closing an Object

When all open views of an object are to be closed, the _wpClose method is invoked. This method is normally invoked when the user selects the *Close* option from a view's context menu.

The _wpClose method may be overridden to perform class-specific processing for closing views, or to free system resources allocated during processing of the _wpOpen method. For example, Figure 176 shows the _wpClose method being overridden to automatically lock a password-protected folder whenever it is closed by the user.

```
SOM_Scope BOOL SOMLINK pwfolder_wpClose(PWFolder *somSelf)
{
    PWFolderData *somThis =                    /* Get instance data     */
                PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",            /* Set debug info        */
                        "pwfolder_wpInitData");
    _LockFolder(somSelf);                      /* Lock folder           */

    return(parent_wpClose(somSelf));           /* Allow default proc    */
}
```

Figure 176. Closing an Object. This example shows the _wpClose method being overridden in order to provide class-specific processing for the password-protected folder.

When a view of an object is closed, the system sends a WM_DESTROY message to the view's frame window. This allows the object to release any allocated resources and save its instance data, so that the object may be reopened in its current state at some future time.

Note that since the _wpClose method is defined by the parent class and is overridden, the default processing performed by the parent is called after the class-specific processing has completed.

14.4.2.5 Saving and Restoring the Object State

As already mentioned, an object is persistent; that is, it remains in existence even when all views of the object are closed. In order to maintain its instance data so that it may subsequently be opened in the same state in which it was closed, the object must save this data when its views are closed and restore it when a view is opened. The Workplace Shell provides methods that handle the saving and restoration of instance data on behalf of object classes; these methods are automatically invoked by the system at the appropriate times, and are described below.

When an object is made dormant, the system invokes the object's _wpSaveState method, which allows the object to save its instance data. A number of predefined methods are available to the object to save its data, such as _wpSaveString. These methods may be called by the object during the processing of its _wpSaveState method, in order to save instance data. An

example of the _wpSaveState method for the password-protected folder example is given in Figure 177 on page 262.

```
SOM_Scope BOOL SOMLINK pwfolder_wpSaveState(PWFolder *somSelf)
{
    PWFolderData *somThis =                    /* Get instance data     */
                 PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",            /* Set debug info        */
                        "pwfolder_wpSaveState");

    _wpSaveString(somSelf,                     /* Save folder password  */
                  "PWFolder",                  /* Class name            */
                   1L,                         /* Class-defined key     */
                   _szPassword);               /* String to be saved    */
    _wpSaveString(somSelf,                     /* Save current password */
                  "PWFolder",                  /* Class name            */
                  2L,                          /* Class-defined key     */
                  _szCurrentPassword);         /* String to be saved    */

    return (parent_wpSaveState(somSelf));      /* Invoke default proc   */
}
```

Figure 177. Saving an Object's State

An object's instance data items are saved in different locations, depending upon the class of the object. Object classes that are descendants of the *WPAbstract* class store their instance data in the OS/2 initialization file OS2.INI. Object classes that are descendants of the *WPFileSystem* class store their instance data in extended attributes. Since the password-protected folder class is descended from the *WPFolder* class defined by the Workplace Shell, which in turn is a descendant of the *WPFileSystem* class, the instance data of this object class is saved as extended attributes in the OS/2 file system.

The class-defined key passed to the _wpSaveString method is used to save the data item in a particular location, which can then be accessed, using the same key, to restore the item. In addition to strings, numeric data may be saved using the _wpSaveLong method, and other binary data such as application-defined data structures may be saved using the _wpSaveData method.

Note that since the _wpSaveState method is defined by the object's class's ancestors and overridden, it must invoke the default processing supplied by the parent class in order to correctly save any instance data defined by ancestor classes. Failure to do so may cause unpredictable results upon awakening the object from its dormant state.

An object must retrieve its instance data and restore its state whenever it is awakened. At this point, the system invokes an object's _wpRestoreState method, which allows the object to restore its state. During the processing of this method, the object can invoke other methods such as _wpRestoreString, which restore specific instance data items. An example of the _wpRestoreState method is given in Figure 178 on page 263.

```
SOM_Scope BOOL32   SOMLINK pwfolder_wpRestoreState(PWFolder *somSelf,
                ULONG ulReserved)
{
    ULONG ulResStrLen;                          /* String length        */

    PWFolderData *somThis =                     /* Get instance data    */
                 PWFolderGetData(somSelf);
    PWFolderMethodDebug("PWFolder",             /* Set debug info       */
                        "pwfolder_wpRestoreState");

    _wpRestoreString(somSelf,                   /* Restore folder p'word */
                     "PWFolder",                /* Class name           */
                     1L,                        /* Class-defined key    */
                     _szPassword,               /* Target string        */
                     &ulResStrLen);             /* Length restored      */
    _wpRestoreString(somSelf,                   /* Restore curr p'word  */
                     "PWFolder",                /* Class name           */
                     2L,                        /* Class-defined key    */
                     _szCurrentPassword,        /* Target string        */
                     &ulResStrLen);             /* Length restored      */

    return(parent_wpRestoreState(somSelf,       /* Invoke default proc  */
                                 ulReserved));
}
```

Figure 178. Restoring an Object's State

The class-defined key passed to the _wpRestoreString method is used to locate the required data item, and the item is restored into the specified target string variable. Numeric data can be restored using the _wpRestoreLong method, and other binary data such as application-defined structures can be restored using the _wpRestoreData method.

Since the _wpRestoreState method is an overridden method, it is important that the default processing supplied by the parent class be invoked. Failure to do so will result in any instance data defined by ancestor classes being in an unknown state, with unpredictable results.

Note that you are not restricted to the workplace methods to save and restore data. You may use any auxiliary file, OS2.INI, extended attributes or whatever means you wish.

14.4.3 Destroying an Object

A specific instance of an object class can be destroyed by the user, simply by dragging it over the Shredder object on the Workplace Shell desktop. If an object or application wishes to delete an object, it may do so using the **WinDestroyObject()** function, as shown in Figure 179.

```
HOBJECT hObject;                            /* Object handle        */
BOOL    bSuccess;                           /* Success flag         */

bSuccess = WinDestroyObject(hObject);       /* Destroy object       */
```

Figure 179. Destroying an Object

The **WinDestroyObject()** function uses the object handle that is returned by the WinCreateObject() function. The object or application that creates the object is responsible for storing this handle during the existence of the object.

When an object is destroyed, the system invokes the object's _wpUnInitData method, which may be used to free any resources or instance data items that were allocated to that particular object.

14.4.4 Deregistering an Object Class

An entire object class can be deleted from the system by deregistering it from the Workplace Shell. This is achieved by either using the **WinDeregisterObjectClass()** function, which is shown in Figure 180, or the **SysDeregisterObjectClass()** function, which is shown in Figure 181.

```
BOOL bSuccess;

bSuccess = WinDeregisterObjectClass(pszClassName);   /* Deregister class */
```

Figure 180. Deregistering an Object Class

The **WinDeregisterObjectClass()** function accepts a string containing the object class name. Once a successful call is made to the **WinDeregisterObjectClass()** function, the object class is deleted from the system and is no longer available to other objects or applications. However, the DLL that contains the code for the object class is not automatically deleted from the system; if the *Templates* folder is subsequently opened with this DLL still resident in a directory in the system's LIBPATH, a template for the class will still appear in the folder. In order to prevent this, the DLL must be explicitly deleted from the system.

During processing of the **WinDeregisterObjectClass()** call, the system invokes the object's _wpclsUnInitData method, to free any instance data or resources that were obtained when the object class was created. See 14.4.1.2, "Class Data" on page 240 for an example of this method.

Figure 181 shows a sample piece of REXX code that deregisters a Workplace Object called pwFinanceFile.

```
/*  */
Call RxFuncadd 'SysLoadFuncs', 'RexxUtil', 'SysLoadFuncs'
Call SysLoadFuncs

'@echo off'

RetCode = SysDeregisterObjectClass( "PWFinanceFile");

if RetCode then
    say 'Uninstall successfully completed for PWFinanceFile class'

say 'Re-boot NOW in order to release DLL'
'pause'
```

Figure 181. REXX Code to Deregister a WPS Object

14.4.5 Accessing PM Resources From a Workplace Shell Object

A Workplace Shell object may access and make use of Presentation Manager resources such as icons, bitmaps, strings and dialogs. These resources may reside in the same DLL as the object's code, or in another DLL. However, since the resources must reside in a DLL, the code that loads the resources must use the **DosLoadModule** or **DosGetModuleHandle()** functions to obtain module handles, as described in Section 9.3.2, "Loading Resources from a DLL", of *OS/2 2.1 Volume 4: Writing Applications*. This is typically done by obtaining the module handles as part of the _wpclsInitData method, and storing them either in global variables or in class data until needed. Note that not all Workplace methods can be assumed to be called in a PM thread. You may need to create your own PM thread to access PM functions during Workplace processing.

14.5 Transient Objects

As mentioned earlier in this chapter, a Workplace Shell object differs from a Presentation Manager window in that it is persistent, that is, it continues to exist after a system restart. The exception to this rule is the **transient object**, which only exists during the current execution of the system, and is destroyed when the system is IPLed.

Transient objects are useful when a programmer wishes to represent and display information such as records from a database. As each record is retrieved, a transient object is created for that record and displayed in a folder on the Workplace Shell desktop. These objects may be opened and manipulated by the end user, but will cease to exist when the system is IPLed.

```
REPLY *Reply;                                   /* Reply data structure  */

SOMAny *NewObj;                                 /* Object pointer        */

case WMP_REQUEST_COMPLETE:
     Reply = (REPLY *)mp1;                      /* Get reply data        */

     ClassID = SOM_IdFromString("CustClass");   /* Get class SOM ID      */

     TransClass = _somFindClass(SOMClassMgrObject, /* Get class pointer  */
                                ClassID,        /* Class SOM ID          */
                                1,1);           /* Major & minor version */

     NewObj = _wpclsNew(TransClass,             /* Create new object     */
                        Reply->CustName,        /* Title for object      */
                        "",                     /* No setup string       */
                        Reply->Folder,          /* Location              */
                        TRUE);                  /* Lock flag             */

     _SetCustInfo(NewObj,                       /* Set instance data     */
                  Reply);
     break;
```

Figure 182. Creating a Transient Object

Figure 182 shows an object window in a requester process which, upon receiving a completed database query from a server process, invokes the

_wpclsNew method to create a new instance of a transient object class, representing the record retrieved from the database.

The SOM pointer to the transient object class is obtained using the **SOM_IdFromString()** macro and the *_somFindClass* method (see 14.6, "Communication between Objects" for further information). This pointer is then used to invoke the _wpclsNew method to create a new instance of the class. Once the new instance is created, a method named *_SetCustInfo*, which is defined by the transient object class, is invoked to insert the information retrieved from the database into the object's instance data.

Note that the technique shown in Figure 182 on page 265 may only be used when an object is created from within the Workplace Shell process. If an object must be created from another process in the system, the WinCreateObject() function must be used.

14.6 Communication between Objects

Objects communicate with one another in order to convey events and initiate actions on the part of other objects. Such communication is typically initiated in one of two ways:

- By an object to convey information to another object with which it has an application-defined relationship, such as a request. This is similar to the *application-initiated* event discussed for Presentation Manager applications in Chapter 4, "The Presentation Manager Application Model", of *OS/2 2.1 Volume 4: Writing Applications*.
- By the user directly manipulating the objects' icons. For example, dropping one icon over another icon initiates a conversation between the two objects. This is similar to the *user-initiated* event discussed for Presentation Manager applications in Chapter 4 of *OS/2 2.1 Volume 4: Writing Applications*.

Each of these types of communication is discussed in the following sections.

14.6.1 Application-Initiated Communication

The application-initiated communication is somewhat more complex than the user-initiated communication since the initiator of the communication typically has knowledge of the type of object to which the communication is being passed, and can usually initiate the communication by simply invoking a method in the receiving object, in a similar manner to that discussed in 14.2.1.9, "Invoking Another Object's Methods" on page 229.

However, it is often necessary to determine the identity of the object for which a method must be invoked. The Workplace Shell provides access to objects using the HOBJECT and the OBJECTID and, at a base level. The system object model provides pointers to objects and SOM IDs. Each of these is described in the following sections, and some discussion is included on converting between identifiers.

14.6.1.1 HOBJECT

This identifier is the object handle, which is allocated by the Workplace Shell and passed as the return value from the WinCreateObject() function. It is a persistent object handle that remains allocated to an object for the duration of its existence. Object handles persist across system restarts, and may therefore be used by one object to refer to another object at any time.

An object handle can be determined from the object's OBJECTID using the **WinQueryObject()** function,

```
HOBJECT  hObject;                              /* Object handle          */
PSZ      szObjectID = "<OBJECTID>";            /* OBJECTID string        */

hObject = WinQueryObject(szObjectID);          /* Query object handle    */
```

Note that this function may be called from any process; its use is not restricted to objects in the Workplace Shell process.

14.6.1.2 OBJECTID

The OBJECTID is provided by an application or object class as part of the setup string parameter in the WinCreateObject() call, when an object is created. It is persistent in the same way as an object handle, but provides a more meaningful reference for an object, which can be used by other objects.

Note that the angle brackets ("<" and ">") used within the OBJECTID are an important part of the syntax.

Note also that the Workplace Shell provides a number of predefined OBJECTIDs for system-defined objects. The first and third WinCreateObject() calls in Figure 183 use the <WP_DESKTOP> OBJECTID to place the objects on the desktop.

```
HOBJECT hObject;                                    /* Object handle        */

/*************************************************************************/
/* Create a folder on the desktop with an OBJECTID of MYFOLDER           */
/*************************************************************************/

hObject = WinCreateObject("WPFolder",           /* Class Name            */
                          "My Folder",          /* Title                 */
                          "OBJECTID=<MYFOLDER>", /* Setup string         */
                          "<WP_DESKTOP>",       /* Location              */
                          CO_REPLACEIFEXISTS);  /* Create option         */

/*************************************************************************/
/* Create a file object with an OBJECTID of MYFILE inside the folder     */
/*************************************************************************/

hObject = WinCreateObject("WPDataFile",         /* Class Name            */
                          "My File",            /* Title                 */
                          "OBJECTID=<MYFILE>",  /* Setup string          */
                          "<MYFOLDER>",         /* Location              */
                          CO_REPLACEIFEXISTS);  /* Create option         */

/*************************************************************************/
/* Create a shadow of the file object MYFILE on the desktop              */
/*************************************************************************/

hObject = WinCreateObject("WPShadow",           /* Class Name            */
                          "My File",            /* Title                 */
                          "SHADOWID=<MYFILE>",  /* Setup string          */
                          "<WP_DESKTOP>",       /* Location              */
                          CO_REPLACEIFEXISTS);  /* Create option         */
```

Figure 183. Referencing an Object Using OBJECTID

14.6.1.3 SOM Pointer

SOM pointers come in various forms, but can all be typecast to *SOMAny **. From a Workplace Shell perspective, a SOM pointer is the return value of the _wpclsNew class method; this is the method used for creating objects within the Workplace Shell process. An object's public methods and data can be accessed using the object's SOM pointer.

A SOM pointer for an object may be obtained from an object handle using the _wpclsQueryObject method provided by the *WPObject* class, as follows:

```
SOMAny *Asomptr;                              /* SOM pointers          */
SOMAny *Bsomptr;

Asomptr = _wpclsQueryObject(_WPObject,        /* Query SOM pointer     */
                            hObject);         /* Object handle         */
```

A SOM pointer for a class may be obtained from the SOM ID for that class, using the *_somFindClass* method shown below:

```
Asomptr = _somFindClass(SOMClassMgrObject,
                        AsomId,
                        1,
                        1);
```

A SOM pointer for a class may be obtained from the SOM pointer for any object within that class, using the *_somGetClass* method as follows:

```
Asomptr = _somGetClass(Bsomptr);
```

The SOM pointer is typically used to invoke class methods from an object in another class. The *_SOMDispatchL()* method shown in Figure 156 on page 229 requires a SOM pointer as a parameter.

14.6.1.4 SOM ID

A SOM ID is simply a way of mapping a unique number to a string. This string may represent the name of a method or class. SOM IDs are integers that are managed by the Workplace Shell using the Atom Manager facility of Presentation Manager. A SOM ID is obtained using the **SOM_IdFromString()** function as follows:

```
somId    AsomId;

AsomId = SOM_IdFromString("WPFolder");
```

The SOM ID is typically used to obtain a SOM pointer, which can then be used to invoke a method.

14.6.2 User-Initiated Communication

The user-initiated communication is somewhat more complex than the application-initiated communication, since the two objects may have no defined relationship. A conversation must be initiated between the two, whereby each determines the nature of the other, and whether a drop operation is valid at the present time. If so, each object passes the information required to carry out the requested action.

14.6.2.1 Dragging a Workplace Object over another Workplace Object

When the user begins to drag an object, this source object being dragged is notified by the system, by invoking the object's _wpFormatDragItem method. This method is used to build a DRAGITEM structure, which is passed to any object if this source object is dragged over it or dropped on it. The DRAGITEM structure contains rendering information about the source object, which is used by other receiving objects over which the source object is dragged, in order to determine whether a drop operation is valid at that point.

Default information for the DRAGITEM structure is inserted by the default processing provided by the parent class, but an object class may override the method and include its own class-specific processing. The DRAGITEM structure is nested within a DRAGINFO structure, which is passed to any receiver object over which one or more source objects are dragged. In a situation where more than one object is being dragged simultaneously, a separate DRAGITEM structure is produced for each source object, and the entire set of structures is combined using a single DRAGINFO structure.

When one object is dragged over another object the system invokes the _wpDragOver method in the receiving object. This method receives a DRAGINFO structure, which contains a variety of information including pointers to one or more DRAGITEM structures. Note it is the responsibility of the receiver object's _wpDragOver method to return whether it can accept the drop, and what operation to perform.

The receiver object has to check the operation code it receives from the source object. If it is the default (*DO_DEFAULT*), it needs to return the operation to be performed, for example *DO_MOVE*, or *DO_COPY*. If it is not the default, for example a *DO_MOVE*, then the receiver object must determine if it can accept that operation and return accordingly.

Warning

If you have written Workplace Shell drag and drop code under OS/2 2.0, the way the ancestor classes respond to the _wpDragOver method has changed for OS/2 2.1. Specifically the parent_wpDragOver method will return *DOR_NEVERDROP* if the ancestor classes cannot accept the source objects.

Consequently code must be written so that the parent_wpDragOver method is called first, and if the resultant Drop Indicator is not *DOR_NEVERDROP*, then the method may continue its processing, as shown in Figure 184 on page 271.

```
SOM_Scope MRESULT   SOMLINK pwFinanceFile_wpDragOver(PWFinanceFile *somSelf,
                HWND hwndCnr,
                PDRAGINFO pdrgInfo)
{
    MRESULT  mr;
    USHORT   dropOperation,
             dropIndicator;
    ULONG    ulItemCount         =0;
    ULONG    ulItem              =0;

    PWFinanceFileData *somThis = PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile","pwFinanceFile_wpDragOver");

    /* firstly will all the source object(s) pass my parents tests? */
    mr = parent_wpDragOver(somSelf,hwndCnr,pdrgInfo);
    dropIndicator = SHORT1FROMMR(mr);
    dropOperation = SHORT2FROMMR(mr);

    if (dropIndicator != DOR_NEVERDROP)
    {
      /* passed the parent's tests, so unless it fails this object's */
      /* tests we will allow the DROP                                */
      dropIndicator = DOR_DROP;
      dropOperation = DO_COPY;

      /* how many items are being dragged ? */
      ulItemCount = DrgQueryDragitemCount(pdrgInfo);

      /* search through the objects and abort if we find any that */
      /* are not WorkPlace objects                                */

      for (ulItem=0; ulItem<ulItemCount; ulItem++) {
        PDRAGITEM pDragItem;               /* temporary variable */
        WPObject *ObjectBeingDragged=NULL; /* temporary variable */

        /* get one of the one or more drag items that we are receiving */
        pDragItem = DrgQueryDragitemPtr(pdrgInfo, ulItem);

        /* test to see if it is a WorkPlace object, if it is use     */
        /* the OBJECT_FROM_PREC macro to get the object; otherwise   */
        /* ObjectBeingDragged will remain as a NULL as it was        */
        /* initialised when declared as NULL                         */
```

Figure 184 (Part 1 of 2). Dragging a Workplace Object

```
        if ( DrgVerifyRMF(pDragItem, "DRM_OBJECT", NULL))
           ObjectBeingDragged = OBJECT_FROM_PREC(pDragItem);
        if (!ObjectBeingDragged) {
            /* Object is NOT a WorkPlace object, so I reject all objects */
            return (MRFROM2SHORT(DOR_NEVERDROP,DO_UNKNOWN));
          } else {
            if ( (pdrgInfo->usOperation != DO_COPY   ) &&
                 (pdrgInfo->usOperation != DO_MOVE   ) &&
                 (pdrgInfo->usOperation != DO_DEFAULT) ) {
            /* this object only allows a move or copy */
            return (MRFROM2SHORT(DOR_NODROP,DO_UNKNOWN));
            } /* endif */
          } /* endif */

        } /* endfor */
    } /* endif */
    /* all the test have been passed, so tell this to the WorkPlace Shell */
    return (MRFROM2SHORT(dropIndicator, dropOperation));
}
```

Figure 184 (Part 2 of 2). Dragging a Workplace Object. In this figure we first see if the source's object passes this object's parent's wpDragOver tests, and then apply our own.

Figure 184 on page 271 shows another _wpDrop method override example for the object called pwFinanceFile which is derived from the wpDataFile Workplace Shell object. In this example we first invoke our objects parent _wpDragOver method (that is the _wpDragOver method for wpDataFile) to see that it doesn't violate any of its rules. If the parent successfully tested one or more source objects, our method determines how many source objects are being dragged over the target object and then searches through each of the source objects() and checks that they are all Workplace Objects. If any one of these is not a Workplace Object then our object will reject all of them by returning *DOR_NEVERDROP* as the indicator and *DO_UNKNOWN* as the operation. Otherwise our method makes sure it is being passed a *DO_COPY*, *DO_MOVE* or a *DO_DEFAULT* operation. If it is not one of these, it will reject all of the source objects. If it is one of these, *DO_DROP* and *DO_COPY* will be returned as the result to the Workplace Shell.

If the receiver object wanted to allow only its tests and not any of the parents, then the parent_wpDragOver method invocation can be removed, along with the setting of the *dropIndication* and *dropOperation* from the returned *mr* program variable, as well as the *if* statement that immediately follows.

If the _wpDragOver in Figure 184 on page 271 returns successfully, and the user drops the object, then the _wpDrop method is invoked for the receiver object. The same checking that was performed in the _wpDragOver method needs to be performed by the _wpDrop method because the receiving object may only receive a _wpDrop method invocation, and not a _wpDragOver.

Note that *DOR_NODROP* is returned when rejecting the operation, and *DOR_NEVERDROP* is returned when rejecting the drop request. This is important when a user drags one object onto another (DO_MOVE) and the receiver object returns a *DOR_NODROP*, meaning it cannot accept the operation. If the user still has the object over the receiver object and now holds down the Ctrl key, then the operation is now a DO_COPY. If the receiver object had previously returned a *DOR_NODROP* then the Workplace Shell will now reinvoke

the objects _wpDragOver method passing it the DO_COPY. This would not occur if a *DOR_NEVERDROP* had been received.

How the receiver object responds with its implementation of the _wpDragOver, depends on:

- The type of object being implemented
- What is being dragged over it
- What the person who designs the object wants to achieve

For example, in Figure 184 on page 271, simple decisions are made based on the parent's _wpDragOver method. Only Workplace Objects which want to perform a *DO_COPY*, *DO_MOVE* or a *DO_DEFAULT* operation are acceptable source objects.

The designer may want to restrict what the object will accept still further, for example by only allowing objects that are the same class as the pwFinanceFile object, (or are descended from it) to be accepted. This could be because there are going to be specific data types that have meaning to this object, or because the designer wishes to have financial account type entries in the pwFinanceFile object and it would therefore not be meaningful to allow the dragging and dropping of a picture or any other objects that could not be understood by this specialized object.

Figure 185 on page 274 expands on Figure 184 on page 271 to now exclude any objects that are not part of the pwFinanceFile class.

```
SOM_Scope MRESULT   SOMLINK pwFinanceFile_wpDragOver(PWFinanceFile *somSelf,
                HWND hwndCnr,
                PDRAGINFO pdrgInfo)
{
    MRESULT  mr;
    USHORT   dropOperation,
             dropIndicator;
    ULONG    ulItemCount         =0;
    ULONG    ulItem              =0;
    CLASS    PWFinanceFileClass;

    PWFinanceFileData *somThis = PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile","pwFinanceFile_wpDragOver");

    /* create a dummy pwFinanceFile class for comparison with the source */
    /* object(s) class                                                   */
    PWFinanceFileClass = _somClassFromId( SOMClassMgrObject,
                         SOM_IdFromString("PWFinanceFile") );

    /* firstly will all the source object(s) pass my parents tests? */
    mr = parent_wpDragOver(somSelf,hwndCnr,pdrgInfo);
    dropIndicator = SHORT1FROMMR(mr);
    dropOperation = SHORT2FROMMR(mr);

    if (dropIndicator != DOR_NEVERDROP)
    {
      /* passed the parent's tests, so unless it fails this object's */
      /* tests we will allow the DROP                                */
      dropIndicator = DOR_DROP;
      dropOperation = DO_COPY;

      /* how many items are being dragged ? */
      ulItemCount = DrgQueryDragitemCount(pdrgInfo);

      /* search through the objects and abort if we find any that */
      /* are not WorkPlace objects                                */

      for (ulItem=0; ulItem<ulItemCount; ulItem++) {
        PDRAGITEM pDragItem;              /* temporary variable */
        SOMAny   *ObjectBeingDragged=NULL; /* temporary variable */

        /* get one of the one or more drag items that we are receiving */
        pDragItem = DrgQueryDragitemPtr(pdrgInfo, ulItem);

        /* test to see if it is a WorkPlace object, if it is use     */
        /* the OBJECT_FROM_PREC macro to get the object; otherwise   */
        /* ObjectBeingDragged will remain as a NULL as it was        */
        /* initialised when declared as NULL                         */

        if ( DrgVerifyRMF(pDragItem, "DRM_OBJECT", NULL))
           ObjectBeingDragged = OBJECT_FROM_PREC(pDragItem);
```

Figure 185 (Part 1 of 2). Only Accepting pwFinanceFile Objects from Drag Operations

```
        if (!ObjectBeingDragged) {
            /* Object is NOT a WorkPlace object, so I reject all objects */
            return (MRFROM2SHORT(DOR_NEVERDROP,DO_UNKNOWN));
          } else {
            if ( (pdrgInfo->usOperation != DO_COPY   ) &&
                 (pdrgInfo->usOperation != DO_MOVE   ) &&
                 (pdrgInfo->usOperation != DO_DEFAULT) ) {
            /* this object only allows a move or copy */
            return (MRFROM2SHORT(DOR_NODROP,DO_UNKNOWN));
            } else {
              /* the object is all ok, but is it a pwFinanceFile object,  */
              /* or descended from one?                                    */
              if (!_somIsA(ObjectBeingDragged,PWFinanceFileClass)) {
                /* reject all the objects because this one is not derived */
                /* from PWFinanceFileClass                                */
                return (MRFROM2SHORT(DOR_NEVERDROP,DO_UNKNOWN));
              } /* endif */

            } /* endif */

          } /* endif */

        } /* endfor */

    } /* endif */
    /* all the test have been passed, so tell this to the WorkPlace Shell */
    return (MRFROM2SHORT(dropIndicator, dropOperation));
}
```

Figure 185 (Part 2 of 2). Only Accepting pwFinanceFile Objects from Drag Operations

14.6.3 Dragging a Non-Workplace Object onto a Workplace Object

Dragging a non-workplace object onto a workplace object is handled in a similar way to the dragging of a workplace object over another workplace object. How the receiver object responds in its implementation of its _wpDragOver depends on the type of object being implemented, what is being dragged over it and what the person who designed the object wants it to do. The difficulty is to determine what the object should do for every possible type of non-Workplace Shell object (and Workplace Shell objects such as OS/2 files).

If the receiver object is capable of having a file object dropped on it, and a file was dragged over it (the source is not a Workplace Object file object), the receiver object could convert the source to a file object and then process it.

If the source of the drag operation cannot be converted to a file object, or is not a file in the first place, then drag and drop participation is still possible if the source object has a drag mechanism and operation that the target object can support. 14.6.5, "Dropping an Object" on page 277 shows an example of a _wpDrop method where an OS/2 file is accepted and converted to a workplace object.

14.6.4 Dragging a Workplace Object onto a Non-Workplace Object

In the same way a non-wrokplace object can participate in a drag and drop conversation with a workplace object, a workplace object can can also participate in a drag and drop conversation with a non-workplace object, provided the workplace object provides support for the drag mechanisms and operations that the receiver object supports.

A workplace object can support multiple mechanisms and operations as shown in Figure 186.

```
MRESULT       mr;
PDRAGTRANSFER pDragTransfer;
BOOL          bSentOK;
   .
   .
   .
/* allocate a drag transfer structure */
pDragTransfer = DrgAllocDragtransfer(1);

if (pDragTransfer) // was the allocate successful?
{
   /* populate the drag structure */
   .
   .
   .
   pDragTransfer->hstrSelectedRMF  =
       DrgAddStrHandle("(DRM_OS2FILE,DRM_PRINT)x(DRF_TEXT),
                        <DRM_OBJECT,DRF_OBJECT>");
   .
   .
   .
   bSentOK = (BOOL)DrgSendTransferMsg(pDragInfo->hwndSource,
                                      DM_RENDERPREPARE,
                                      (MPARAM)pDragTransfer,
                                      (MPARAM)NULL);
   if (bSentOK)
   {
     .
     .
     .
   } else {
     mr = (MRESULT)RC_DROP_ERROR;
   }
     .
     .
     .
}
```

Figure 186. Multiple Rendering Methods

14.6.5 Dropping an Object

When a drop operation occurs, the receiver object is notified by the system which invokes the _wpDrop method for that object. This method accepts the DRAGINFO structure which may then be examined by the receiver object to determine the correct action to be taken. The rendering information contained in the DRAGITEM structure may be sufficient to allow the action to be completed, or the receiver object may initiate a conversation with the source object in order to gain sufficient information to complete the action.

If the source object is not a workplace object but it is an OS/2 file, then the receiver object must decide whether it can handle a file. If receiver handling is set then the receiver object can create a workplace object created to represent the OS/2 file. This workplace object can then issue methods. However, if you do not wish to create a workplace object to represent the file then the Presentation Manager Drag and Drop messages must be handled.

If neither a workplace object, nor an OS/2 file is being dropped, then the receiver object must decide what it wants to do.

The rendering information provided in the DRAGITEM structure, and its use by a Presentation Manager or Workplace Shell object, are described in detail in Chapter 8, "Direct Manipulation" of *OS/2 2.1 Volume 4: Writing Applications*.

```
/*
 *
 *   METHOD:   wpDrop                                          PUBLIC
 *
 *   PURPOSE:  To receive a dropped object.
 *
 *   INVOKED:  By Workplace Shell, when another object has been dropped on
 *             this object.
 *
 */

SOM_Scope MRESULT   SOMLINK pwFinanceFile_wpDrop(PWFinanceFile *somSelf,
          HWND hwndCnr,
          PDRAGINFO pdrgInfo,
          PDRAGITEM pdrgItem)
{

    CHAR          szName[CCHMAXPATH];
    CHAR          szPath[CCHMAXPATH];
    ULONG         cbPath = CCHMAXPATH;
    ULONG         ulItemCount         =0;
    ULONG         ulItem              =0;
    CLASS         PWFinanceFileClass;
    SOMAny       *ObjectBeingDragged;
    MRESULT       mr;
    USHORT        dropOperation,
                  dropIndicator;
    BOOL          flPrepared = TRUE; // Assume we do not need to do a prepare
    BOOL          flRendering = FALSE;
    PDRAGTRANSFER pDragTransfer;

    PWFinanceFileData *somThis = PWFinanceFileGetData(somSelf);
    PWFinanceFileMethodDebug("PWFinanceFile","pwFinanceFile_wpDrop");

    if ((strcmp(_szCurrentPassword,       /* If FinanceFile is NOT locked */
                _szPassword)) == 0)
    {

  /* make sure we are not dragging ourselves, and dropping onto ourselves */
     if (pdrgInfo->hwndSource != hwndCnr)
     {

     /* for each of the items being dropped, check to see that they are all */
     /* derived from PWFinanceFileClass                                     */

     PWFinanceFileClass = _somClassFromId( SOMClassMgrObject,
                          SOM_IdFromString("PWFinanceFile") );
```

Figure 187 (Part 1 of 5). Converting a Source Drag OS/2 File to a Workplace Object

```
/* passed the parent's tests, so unless it fails this object's */
/* tests we will allow the DROP                                */
dropIndicator = DOR_DROP;
dropOperation = DO_COPY;

/* how many items are being dragged ? */
ulItemCount = DrgQueryDragitemCount(pdrgInfo);

/* search through the objects and abort if we find any that aren't derived */
/* from PWFinanceFileClass                                                 */
somPrintf("Number of Items being dropped = %i.\n",ulItemCount);
for (ulItem=0; ulItem<ulItemCount; ulItem++) {
  PDRAGITEM pDragItem; /* temporary variable*/

  /* get one of the one or more drag items that we are receiving */
  pDragItem = DrgQueryDragitemPtr(pdrgInfo, ulItem);

  ObjectBeingDragged = queryObjectFromDragItem(pDragItem);

  if (ObjectBeingDragged) {
    if (!_somIsA(ObjectBeingDragged,PWFinanceFileClass)) {
      somPrintf("Object %i, is rejected for drop because it ",ulItem);
      somPrintf("is not derived from PWFinanceFileClass\n");
    } else {
      somPrintf("Object %i, is acceptable for dropping, by wpDROP\n",ulItem);
    } /* endif */
  } else {
    somPrintf("Object %i, is not a WPS object, can we render it\n",ulItem);

/* start of code to render item */

if( DrgVerifyRMF (pDragItem, "DRM_OS2FILE", NULL) )
{
   somPrintf("An OS2FILE rendering method!\n");
   /* Protocol allows the source object to propose a target name...
    *
    * If it does, then try to use it, if it does not, then
    * try to use the source name, if present.  Finally, just
    * make up our own name...
    */
   if (pDragItem->hstrTargetName &&
       DrgQueryStrNameLen(pDragItem->hstrTargetName) )
   {
      DrgQueryStrName(pDragItem->hstrTargetName,
                      sizeof(szName),szName);
      somPrintf("Source proposes the target filename\n");
   }
   else
   {
```

Figure 187 (Part 2 of 5). Converting a Source Drag OS/2 File to a Workplace Object

```
      if (pDragItem->hstrSourceName &&
         DrgQueryStrNameLen(pDragItem->hstrSourceName))
      {
         DrgQueryStrName(pDragItem->hstrSourceName,
                         sizeof(szName),szName);
      somPrintf("Source proposes the source filename\n");

      }
      else
      {
         szName[0] = '\0';
         somPrintf("no source, nor target name\n");
      }
   }

   /* Allocate and initialize a drag transfer structure
    */
   somPrintf("allocating pDragtransfer structure\n");
   pDragTransfer = DrgAllocDragtransfer(1);

   if (pDragTransfer)
   {
      somPrintf("pDragtransfer structure allocated ok\n");
      /* create a WPDataFile object
       */
      ObjectBeingDragged = _wpclsNew( _WPDataFile,
                          szName,
                          NULL,
                          _wpclsQueryFolder(_WPDataFile,"<WP_NOWHERE>",TRUE),
                          TRUE );

      if (ObjectBeingDragged)
      {
         somPrintf("ObjectBeingDragged has been successfully allocated\n");
         _wpQueryRealName(ObjectBeingDragged,szPath,&cbPath,TRUE);
         somPrintf("The ObjectBeingDragged filename is %s\n",szPath);

         /* fill in the struct now
          */
         pDragTransfer->cb               = sizeof(DRAGTRANSFER);
         pDragTransfer->hwndClient       = hwndCnr;
         pDragTransfer->pditem           = pDragItem;
         pDragTransfer->hstrSelectedRMF  =
                        DrgAddStrHandle("<DRM_OS2FILE,DRF_UNKNOWN>");
         pDragTransfer->hstrRenderToName = DrgAddStrHandle( szPath );
         pDragTransfer->ulTargetInfo     = 0L;
         pDragTransfer->usOperation      = pdrgInfo->usOperation;
         pDragTransfer->fsReply          = 0;
```

Figure 187 (Part 3 of 5). Converting a Source Drag OS/2 File to a Workplace Object

```
/* Now, if the source wants prepared, do it...
 */
if (pDragItem->fsControl & DC_PREPARE)
{
   somPrintf("Source wants prepared\n");
   flPrepared = (BOOL)DrgSendTransferMsg(pdrgInfo->hwndSource,
                                         DM_RENDERPREPARE,
                                         (MPARAM)pDragTransfer,
                                         (MPARAM)NULL);
} else {
   somPrintf("Source does not want prepared\n");
}
/* See if either we did not need to send a RENDERPREPARE, or
 * we have successfully done so...
 */

if (flPrepared)
{
   somPrintf("not prepared\n");
   /* Tell the source object where to put the file.
    */
   flRendering = (BOOL)DrgSendTransferMsg(pDragItem->hwndItem,
                                          DM_RENDER,
                                          (MPARAM)pDragTransfer,
                                          (MPARAM)NULL);
   if (!flRendering)
   {
      /* The partner object did not render, so delete
       * the object we just created.
       * or we could add code here to directly open the source as
       * a file and work with it, or what ever we like.
       */

      _wpFree(ObjectBeingDragged);
      somPrintf("not rendering, we are deleting the object we just created\n");
   } else {
      somPrintf("rendering\n");
   }
}
else
{
```

Figure 187 (Part 4 of 5). Converting a Source Drag OS/2 File to a Workplace Object

```
                somPrintf("Our partner wanted us to send him a prepare, and");
                somPrintf("now has changed his mind about things..., ABORT\n");

                /* Our partner wanted us to send him a prepare, and
                 * now has changed his mind about things...
                 * We cannot even send him an end conversation, as
                 * we do not know that the hwnd is any good.
                 *
                 * For now, we will treat this as an error.
                 */
                mr = (MRESULT)RC_DROP_ERROR;
              }
            } else {
              somPrintf("ObjectBeingDragged has NOT been successfully allocated\n");
            }
            if (flRendering)
            {
               mr = RC_DROP_RENDERING;
            }
            else
            {
               DrgDeleteStrHandle( pDragTransfer->hstrRenderToName );
               DrgFreeDragtransfer( pDragTransfer );
            }
          }
        } else {
           somPrintf("Not an OS2FILE rendering method\n");
        }

          } /* endif */

        } /* for */
      } else {
         somPrintf("we are trying to drop onto ourselves, not allowed\n");
      } /* endif */
    } else {
       somPrintf("LOCKED, drop is disallowed\n");
    } /* endif */

    return((MRESULT) NULL);

}
```

Figure 187 (Part 5 of 5). Converting a Source Drag OS/2 File to a Workplace Object. A partial sample _wpDrop accepting a non-Workplace Object, specifically an OS/2 file.

14.7 Building a Workplace Shell Application

As already mentioned, an application that exploits the Workplace Shell consists of a number of objects on the desktop or in folders, which interact with one another to carry out operations as requested by the user. The implementation of the Workplace Shell under OS/2 2.0 causes all Workplace Shell objects to run in a single process, under the control of the Workplace Shell itself. It is therefore possible for an error in a Workplace Shell to terminate the Workplace Shell process, and all objects currently open under the control of that process. While the Workplace Shell automatically restarts itself and its open objects, it is recommended for reasons of performance that applications carrying out lengthy processing such as database or remote system access should be implemented using multiple processes. Other processes in the system are not affected if the Workplace Shell process terminates, and become available to the user as soon as the shell restarts itself, without the need to reload application code, reinitialize communications links, etc.

For example, a database query application that searches a database for customer records and displays these in a Workplace Shell folder may be composed of two processes, each with multiple threads, as shown in Figure 188 on page 283.

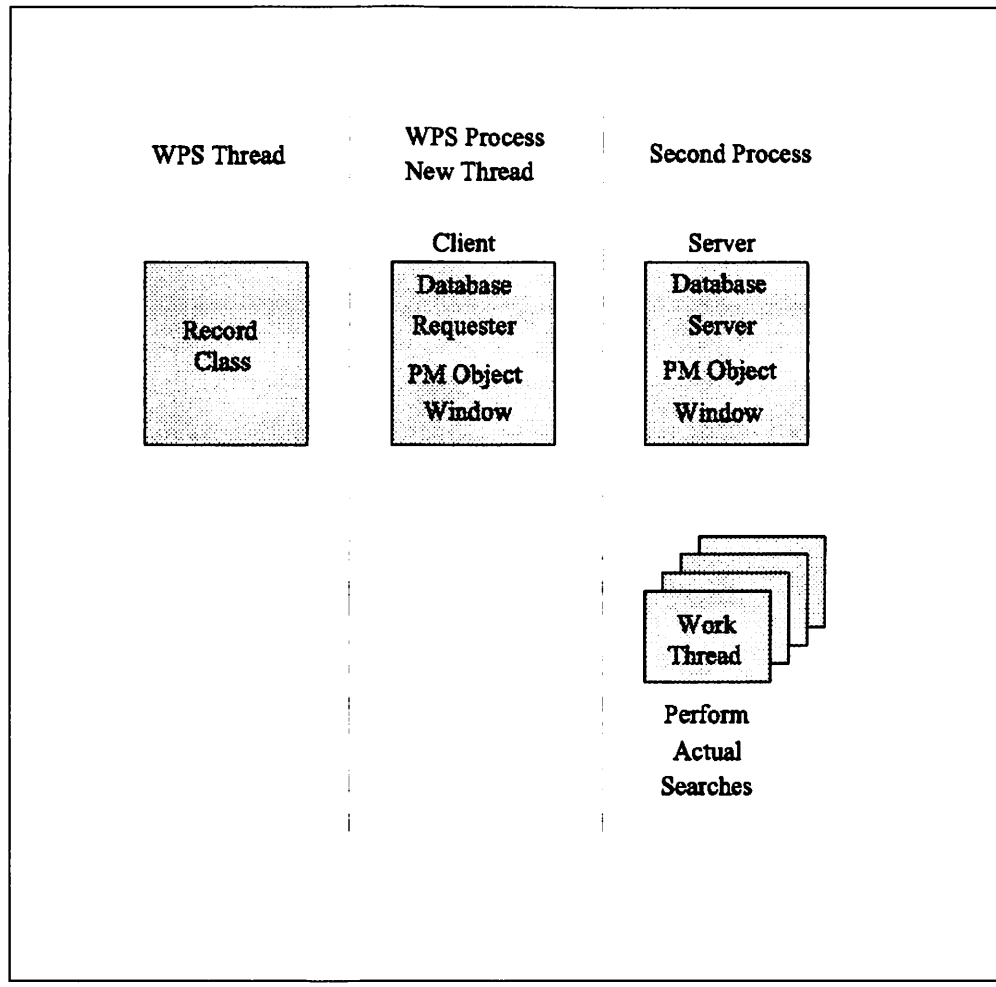


Figure 188. Possible Workplace Shell Application Structure

The requester portion of the application, which allows the user to enter a query, and which displays the results on the screen, is implemented as a Workplace Shell object, running under the control of the Workplace Shell process. The primary thread in this process carries out the interaction with the end user, while a secondary thread is created to handle communication between processes.

The second process acts as a database server, and is created by the first process when the application is started. The server process has a primary thread that accepts requests from the requester in the Workplace Shell process, and a number of secondary threads that actually perform the database access.

If an errant object or application were to cause the Workplace Shell to terminate, the requester threads would be terminated. However, the server process *would not* be terminated, and communication with the requester could be re-established simply by having the requester initiate one of the standard interprocess communication mechanisms described in *OS/2 2.1 Volume 4: Writing Applications, Chapter 10, "Multitasking Considerations"*.

14.8 Debugging

SOM provides several facilities to aid in debugging SOM and WPS applications. These facilities are designed around a replaceable procedure called *SOMOutCharRoutine*, which normally writes output to the *stdout* logical device.

It is not practical to capture *stdout* and the information that SOM and WPS is providing when debugging WPS objects, because objects are implemented as DLLs. Instead the object must replace the *SOMOutCharRoutine* to send the output to a place where you can easily deal with it.

Additionally the *OS/2 Programmer's Toolkit* provides an interactive debugging tool, the Kernel Debugger. For further information on using the Toolkit Kernel Debugger to debug DLLs, please refer to the online documentation for the *OS/2 Programmer's Toolkit*.

14.8.1 Replacing SOM's SOMOutCharRoutine

Figure 189 on page 285 shows a simple replacement procedure that sends the output to the *COM1:* serial port. To actually replace the *SOMCharOutRoutine* it is necessary to add a line of code to the initialization portion of your object. This is shown in Figure 190 on page 285. Note that it is best to add a command to your *STARTUP.CMD* file to set up the *COM1:* serial port. A sample portion of a *STARTUP.CMD* is shown in Figure 191 on page 286.

You can now use another computer to view the information that is generated by the SOM runtime as you manipulate your object.

The additional computer should be connected from its serial port, normally COM1, to the first computer by means of a NULL modem cable. This is a specialized serial cable where the transmit and receive conductors are crossed over to allow the transmission from one serial port to be received by the other.

It is then a matter of registering your object and running an ASCII terminal emulator program on the additional computer to view the information.

```
    .
    .
    .
#
# Passthru a debug message box to the .ih file
# (for inclusion in the .c file)
#
passthru: C.ih, after;

#include <wppgm.h>
#include <wppgmf.h>
#include <stdio.h>

// force SOM to output all debug information to the Communications Port 1

int myReplacementForSOMOutCharRoutine (char c)
{
static FILE *fdebug = NULL;

if (!fdebug) {
  fdebug = fopen("COM1","w");

  if (!fdebug) return 0;     /* failed to open COM1: */
  }
fputc(c,fdebug);
fflush(fdebug);

return 1;
}
endpassthru;
    .
    .
    .
```

Figure 189. Sample .CSC File Definition for Overriding the SOMOutCharRoutine

```
    .
    .
    .
    /* Set up the debug and tracing ... */

    /* Produce a message each time a method is entered */
    SOM_TraceLevel=2;

    /* Replace the default routine with this object's new one */
    SOMOutCharRoutine = myReplacementForSOMOutCharRoutine;
    .
    .
    .
```

Figure 190. Sample .C File Definition for Overriding the SOMOutCharRoutine

```
/* My STARTUP.CMD */
MODE COM1 9600,n,8,1
/* and whatever else I like to have in here */
    .
    .
    .
exit 0
```

Figure 191. Sample STARTUP.CMD File Definition

14.8.2 A Sample ASCII Terminal Emulator for Debugging Use

The PM terminal program that can be found in the Productivity folder can be used to receive and display the information from the SOM runtime about the computer to be debugged. The following steps detail how to create and use a custom emulator session.

14.8.2.1 Creating a Custom Emulator Session

The following steps detail how to create a PM Terminal session suitable for remote debugging use.

1. Open **OS/2 System** and select the **Productivity** Folder.
2. Select **PM Terminal**.
3. From the **Session** pull-down, select **Add**.
4. Enter a comment, for example "Remote SOM Debug Session".
5. You should have the default settings as shown in Table 27, on the **Add Session** panel.
6. Select the **ADD** push button to add the new session.

Table 27. Parameters and Settings for the Remote Terminal

Parameter	Setting
Terminal emulation profile	ANSI 3.64
Connection path profile	ACDI - Hardwire
System environment profile	Default Environment
File transfer profile	Character

14.8.2.2 Using the Emulator Session

After you have created an instance of your object, start the session you created above. If the connection is broken at any time, for example you closed and reopened the object, then from the **File** pulldown select **Connect**.

14.8.3 SOM Provided Macros for Debugging

System Object Model provides a number of macros for the purposes of debugging objects. These are:

SOM_TestC Evalutes a Boolean expression. If it is true, then execution continues; otherwise **SOM_Error** is invoked.

SOM_WarnMsg Writes out a warning message depending on the setting of the SOM_WarnLevel variable.

SOM_Assert Evaluates a Boolean assertion. If this fails then SOM_Error is invoked with a user-supplied error code.

SOM_Expect	Evaluates a Boolean assertion. If this fails then SOM_Warn is invoked.
somPrintf	SOM's implementation of the "C" printf function.

For more detailed information please refer to *System Object Model Guide and Reference*.

14.9 Summary

The OS/2 Version 2.0 Workplace Shell provides an object-oriented user interface to the operating system, and provides an object-oriented application layer on top of Presentation Manager, through its implementation of the system object model. An application that exploits the facilities of the Workplace Shell consists of a number of objects, which are manipulated on the Workplace Shell desktop by the user, in order to carry out the required tasks.

Workplace Shell objects conform to "standard" object-oriented principles in that they are grouped into object classes, have instance data, and contain methods which perform the required tasks and operate upon the instance data. Workplace Shell object classes may inherit data and methods from their parent class, in accordance with the object-oriented concept of inheritance. A class may add additional data items or new methods to perform actions not handled by its parent class, or may override existing methods to handle actions in a different manner.

An object class is defined using a class definition file, which defines the parent hierarchy for the object, its data items and its methods. The class definition file is used as input to the SOM Precompiler, which uses the file to produce a number of source code files and header files. The source code is edited by the programmer to add the application logic for each method. It is then compiled using a normal C compiler, and link edited to produce a dynamic link library that implements the object class. An object class may make use of operating system and Presentation Manager resources during its execution.

Workplace Shell objects behave in a similar manner to windows under Presentation Manager; object classes are registered to the Workplace Shell, and individual instances of an object class are created, opened, closed and destroyed by the user or by other objects in the system. The major difference between a window under Presentation Manager and an object under the Workplace Shell is that Workplace Shell objects are persistent, that is, they exist for longer than a single application execution. Once created, a Workplace Shell object remains in existence until it is explicitly destroyed.

Appendix A. OS/2 2.00.1 and Service Pak XR06055 Enhancements

Since OS/2 2.0 was first released in March 1992, IBM has provided new and improved functions within the product, culminating in the release of OS/2 2.1.

There have been a number of intermediate releases of OS/2 Version 2 between OS/2 2.0 and OS/2 2.1, and the aim of this chapter is to explain what the different releases are, and to list which enhancements are included.

A.1 Release Summary

The first part of this chapter defines the various release of OS/2 Version 2, describes the major highlights of each release, and discusses their interrelationship.

OS/2 2.0 was released in March/April 1992.

This was available through various means:

- As a shrinkwrap package containing documentation and either 3.5" or 5.25" diskettes.
- As an upgrade from DOS, Windows or OS/2 1.x. This is a shrinkwrap package containing the documentation and diskettes, but also containing a "sniffer" program to check that this is an upgrade rather than a fresh install
- Preloaded onto selected IBM PS/2 models.

OS/2 2.00.1 became available in September/October 1992.

It was released in order to support the IBM PS/2 and PS/ValuePoint systems announced at that time. It included both enhancements and bug fixes (APARs). OS/2 2.00.1 has also been referred to sometimes as OS/2 V2.01. The terms **OS/2 2.00.1** and **OS/2 V2.01** are synonymous, and refer to the same product.

OS/2 2.00.1 was only available preloaded or as backup to preloaded systems:

- Preloaded OS/2 2.00.1 for the PS/2 Models 56, 57, 76, 77, and new versions of the PS/2 Model 90
- Preloaded OS/2 2.00.1 for the PS/ValuePoint systems
- OS/2 2.00.1 Backup Diskettes, which could be ordered by telephone on a per-country basis for a nominal, cost-recovery fee, in order to restore OS/2 2.00.1 onto these systems

A public beta-test version of OS/2 2.00.1 was released in September 1992, along with a public beta-test version of WIN-OS/2 3.1. This should never have been used for production purposes, and has now been superseded by OS/2 2.1.

Service Pak XR06055 for OS/2 2.0 was released electronically on IBM and external bulletin boards in October/November 1992.

The Service Pak XR06055 diskettes could also be ordered by telephone for a nominal, cost-recovery fee. OS/2 2.0 with Service Pak XR06055 applied provided a level of OS/2 Version 2 slightly newer than OS/2 2.00.1; the main difference was that additional bug fixes (APARs) were included in the Service Pak XR06055.

OS/2 2.1 was announced in May 1993, and became generally available shortly afterwards.

Public beta-test versions of OS/2 2.1 were released during late 1992 and early 1993. The license agreement for these beta-test versions has expired with the general availability of OS/2 2.1, and any remaining copies of the beta-test versions should now be replaced by licensed copies of OS/2 2.1.

A.2 PS/2, PS/VP, ThinkPad and PS/1 Hardware Support

The following table can be used to check which version of OS/2 Version 2 will work on which models of recently announced IBM hardware.

For information on OS/2 2.1 support for IBM systems, refer to section 4.11, "Preloaded IBM Hardware Systems" on page 60.

Table 28. IBM PS Support for OS/2 2.0, OS/2 2.00.1 and Service Pak XR06055 for OS/2 2.0

Machine Type and Model	OS/2 2.0	Preloaded OS/2 2.00.1	OS/2 2.0 with Service Pak XR06055 applied
PS/2 Models 9556, 9557	4,5	Y	4,5
PS/2 Models 9576, 9577	1	Y	Y
PS/2 Server 9585	4,5		1
PS/2 Model 9590	1	Y	Y
PS/2 Model 9595	1		Y
PS/VP 1 325T	2		2
PS/VP 1 425SX	2	Y	Y
PS/VP 1 433DX	2	Y	Y
PS/VP 1 466DX2	2	Y	Y
PS/VP 2 Model 6382	2	Y	Y
PS/VP 2 Model 6384	2	Y	Y
PS/VP 2 Model 6387	2	Y	Y
ThinkPad 300	2		2
ThinkPad 700, 700C, 720, 720C	4,6	3	4,6
PS/1 386SX	2		2
PS/1 486SX	2		Y

Table Notes

1. New functions not exploited are ISO fonts, 2.88MB diskette lock/unlock, read/write optical drive lock/unlock.
2. VGA mode only.
3. Special OS/2 2.00.1 Backup Pak for ThinkPad 700 series needed - order through the number in the ThinkPad package
4. If you have not already done so, create a backup copy of your system partition following the instructions supplied with your system. At this time you MUST make copies of your OS/2 "Install" and "Diskette 1" diskettes. Please label these diskettes "9556/57 Install" and "9556/57 Diskette 1" or 9585 or ThinkPad 700 depending on your system.
5. If you have a 9556 or 9557, copy the ABIOS.SYS and SF838XX.BIO files from the backup copy of the reference diskette to the OS/2 2.0 "9556/57 Install" and "9555/57 Diskette 1" diskettes you created in Note 4. If you have a 9585, copy the ABIOS.SYS and SF848XX.BIO files from the backup copy of the reference diskette to the OS/2 2.0 "9585 Install" and the "9585 Diskette 1" diskettes.
6. If you have a Thinkpad 700, copy your hardware system *.BIO file to INSTALL and Diskette 1. Modify your system ABIOS.SYS file to include your *.BIO file name as the first entry in ABIOS.SYS file.

A.3 Packaging Media Details for OS/2 Version 2

OS/2 Version 2 is provided on 3.5" diskettes, 5.25" diskettes, and for OS/2 2.1 on CD-ROM (with accompanying installation diskettes).

As a check, here is a summary of the number of diskettes provided in the 3.5" diskette and CD-ROM packages for each version of OS/2 Version 2:

- **Install** is the initial OS/2 installation diskette.
- **Product** is the remaining OS/2 diskettes, numbered sequentially.
- **Printer Drivers**, also called Device Drivers, is the set of diskettes containing the OS/2 printer drivers.
- **Display Drivers** is the set of diskettes containing the OS/2 display drivers.
- **MMPM/2** is the set of diskettes for the MultiMedia Presentation Manager/2* extensions to OS/2.
- **Preload Install** is the set of diskettes specfically for reinstalling a preloaded system, and contains some extra utilities.
- **CD-ROM** is the CD-ROM which is equivalent to the OS/2, printer driver, display driver and MMPM/2 diskettes.
- **3.5" Install** and **5.25" Install** are the two installation diskettes needed to start installation from CD-ROM.

Table 29. OS/2 Version 2 3.5" Diskette Count

OS/2 Version 2 Version	Install	Product	Printer Drivers	Display Drivers	MMPM/2 Product	Preload Install
OS/2 2.0 Shrinkwrap	1	15	5			
OS/2 2.0 Preload Backup	1	15	5			2
OS/2 2.00.1 Preload Backup	1	15	5	3		2
Service Pak XR06055 for OS/2 2.0		12		2		
OS/2 2.1 Shrinkwrap	1	17	3	2	2	
OS/2 2.1 Preload Backup	1	17	3	2	2	2(US), 3(EMEA)

Table 30. OS/2 Version 2 CD-ROM and Diskettes Count

OS/2 Version 2 Version	CD-ROM	3.5" Install	5.25" Install
OS/2 2.1 CD-ROM Shrinkwrap	1	2	2

A.4 How to Check which Version of OS/2 2.x Is Installed

With all these various versions and releases of OS/2 2.x being available, it is often useful to be able to check which version is installed.

Our recommendation is to use the **SYSLEVEL** command from an OS/2 command prompt in either a windowed or full-screen OS/2 session.

The output from SYSLEVEL is line-mode and is displayed one page at a time. SYSLEVEL will report on all the OS/2 system components installed on all the disks in your system. You are looking for two pieces of information - the OS/2 Base Operating System level, and the OS/2 Graphics Engine level. If you have more than one version of OS/2 installed on your system, you will get this information reported for each version; in this case look for the information for the disk from which you booted OS/2.

Non-US systems will have a country letter as the third character of the CSD level; for example, the UK CSD level for OS/2 2.0 is XRU2000.

```
C:\OS2\INSTALL\SYSLEVEL.OS2
IBM OS/2 Base Operating System
Version 2.00     Component ID 562107701
Type 0
Current CSD level: XR02000
Prior   CSD level: XR00000

C:\OS2\INSTALL\SYSLEVEL.GRE
IBM OS/2 16-bit Graphics Engine
Version 2.00     Component ID 562107701
Type 0
Current CSD level: XR02000
Prior   CSD level: XR00000
```

Figure 192. OS/2 2.0 - SYSLEVEL Output

```
C:\OS2\INSTALL\SYSLEVEL.OS2
IBM OS/2 Base Operating System
Version 2.00.1     Component ID 562107701
Type 0
Current CSD level: XR02010
Prior   CSD level: XR02000

C:\OS2\INSTALL\SYSLEVEL.GRE
IBM OS/2 32-bit Graphics Engine
Version 2.01.1     Component ID 562107701
Type 0
Current CSD level: XR02010
Prior   CSD level: XR02000
```

Figure 193. OS/2 2.00.1 - SYSLEVEL Output

```
C:\OS2\INSTALL\SYSLEVEL.OS2
IBM OS/2 Base Operating System
Version 2.00     Component ID 562107701
Type 0
Current CSD level: XR06055
Prior   CSD level: XR02000

C:\OS2\INSTALL\SYSLEVEL.GRE
IBM OS/2 32-bit Graphics Engine
Version 2.01.1     Component ID 562107701
Type 0
Current CSD level: XR02010
Prior   CSD level: XR02000
```

Figure 194. OS/2 V2.0 with Service Pak XR06055 Applied - SYSLEVEL Output

```
C:\OS2\INSTALL\SYSLEVEL.OS2
IBM OS/2 Base Operating System
Version 2.10     Component ID 562107701
Type 0
Current CSD level: XR02010
Prior   CSD level: XR02010

C:\OS2\INSTALL\SYSLEVEL.GRE
IBM OS/2 32-bit Graphics Engine
Version 2.10     Component ID 562107701
Type 0
Current CSD level: XR02010
Prior   CSD level: XR02010
```

Figure 195. OS/2 2.1 - SYSLEVEL Output

A.5 OS/2 2.00.1

OS/2 2.00.1 is an enhanced version of OS/2 2.0 which contains both enhancements and APAR fixes.

The APAR fixes in OS/2 2.00.1 are listed in Appendix D, "APAR Fixes in OS/2 2.00.1, Service Pak XR06055, and OS/2 2.1" on page 341.

OS/2 2.00.1 was preloaded onto selected IBM PS/2 and PS/ValuePoint systems, and also onto some Dell system.

The benefit of preloading OS/2 2.00.1 is that when the customer receives his machine, he can simply turn on the machine and start using the system immediately.

OS/2 2.00.1 was only available in preloaded form. It is possible, using Preloaded OS/2 2.00.1, to install or remove system features without having to use diskettes. For this reason the customer is not given any diskettes when they receive their machine. There is, however, documentation that accompanies the machine which explains how the customer can create their own backup copy of Preloaded OS/2 2.00.1. As an alternative, the customer can order a backup copy of the diskettes from IBM or their dealer.

Preloaded versions of OS/2 Version 2, including Preloaded OS/2 2.00.1, also contain the Welcome Folder and additional utilities.

A.5.1 OS/2 2.00.1 Reinstallation

OS/2 2.00.1 was preloaded onto new IBM PS/2 and PS/ValuePoint systems. In addition, a Welcome Folder and some useful utilities were installed onto the preloaded systems.

If it is necessary to reload OS/2 Version 2 onto these systems, there are three alternatives:

- Backup the preloaded OS/2 2.00.1 to diskettes when first purchased, using the BACKUP.EXE located on the utility diskettes that the user must create. Then restore the system using the RESTORE.EXE located on the utility diskettes. The system must be booted with these utility diskettes to do both

the backup and the restore. This is so that the complete system can be backed up and then restored properly.

Note that the use of OS/2 2.0 on 9556 and 9557 machines requires the presence of the SF838XX.BIO file for installation purposes.

Note also that, if you have a set of backup diskettes, after installation you may not want to restore the Welcome Folder utilities since the programs located in the Configuration Tools folder inside the Welcome Folder were put there because you did not have diskettes in the first place. It becomes a question of what you want to do with them.

- Order the OS/2 2.00.1 Preload Backup Diskettes (these can be ordered by telephone on a per-country basis, for a nominal media-recovery charge)
- Restore OS/2 2.0 onto the system, configure the XGA or SVGA display if necessary, and apply the Service Pak XR06055 for OS/2 2.0.

The Service Pak XR06055 for OS/2 2.0 can now be applied to OS/2 2.00.1.

Special considerations apply to installing OS/2 2.0 and the Service Pak XR06055 onto certain systems. See A.2, "PS/2, PS/VP, ThinkPad and PS/1 Hardware Support" on page 290 for details.

A.6 Service Pak XR06055 for OS/2 2.0

The Service Pak XR06055 for OS/2 2.0 was provided to enable all the remaining users of OS/2 2.0 to benefit from the enhancements and bug fixes in OS/2 2.00.1. Since it was produced at a later date than Preloaded OS/2 2.00.1, additional bug fixes were ready to be included, and thus the level of OS/2 2.0 with Service Pak XR06055 applied is slightly greater than Preloaded OS/2 2.00.1.

The Service Pak contains all of the enhancements and bug fixes in OS/2 2.00.1, along with additional fixes. It is applied only to OS/2 2.0. It should not be applied to any versions of OS/2 2.00.1.

IBM has freely distributed the Service Pak through various different information networks such as:

- CompuServe**
- Public and private bulletin boards
- IBM's own OS/2 bulletin board service
- IBM Support Centers

The diskettes can be ordered from IBM for a nominal fee, which covers the cost of media, copying and shipping. Otherwise, the Service Pak is a no-charge offering and can be freely distributed to other users.

The Service Pak XR06055 contains all the enhancements and almost all of the APAR fixes contained in OS/2 2.00.1, along with additional APAR fixes. For exact details, refer to Appendix D, "APAR Fixes in OS/2 2.00.1, Service Pak XR06055, and OS/2 2.1" on page 341. A complete list of the APARs in the Service Pak XR06055 can also be found in a file called "SPAPAR.TXT" which is included with the Service Pak XR06055.

Note

Although the Service Pak XR06055 has solved many bugs, our recommendation is for all OS/2 Version 2 users to upgrade to OS/2 2.1. This provides the latest and most stable platform for the future.

APAR fixes will continue to be provided for users continuing on the OS/2 2.0 with Service Pak XR06055 applied base.

A.6.1 Pre-Installation Considerations

Prior to installing the Service Pak XR06055 for OS/2 2.0 there are some key points you should consider:

- The Service Pak XR06055 must *not* be installed on Preloaded OS/2 2.00.1 machines, since these machines have a slightly different version of OS/2 Version 2 which is not compatible with the Service Pak XR06055. Many of the fixes that are in the Service Pak XR06055 are already contained in OS/2 2.00.1.
- The Service Pak XR06055 contains some new printer drivers that replace some of the OS/2 2.0 and OS/2 1.3 printer drivers. Some of these drivers will be updated by the Service Pak XR06055, while others will have to be updated by you. If the drivers are not updated then you may not be able to print when the system is restarted. A list of the drivers that have been updated can be found in Chapter 10, "Print Subsystem Enhancements" on page 129.
- If you plan to add any additional system features like additional printer drivers or any of the selective install features, these features should be installed prior to installing the Service Pak XR06055. Adding any features after the Service Pak XR06055 has been installed may require you to re-install the Service Pak XR06055.
- The new code which is contained in the Service Pak XR06055 can take up to 3MB of extra disk storage depending on the currently installed features. Before installing the Service Pak XR06055 make sure there is sufficient space on the hard disk to accommodate the extra 3MB of disk space required. One way to provide more space on the hard disk is by moving the SWAPPER.DAT file to another logical drive.
- If the Advanced version of OS/2 LAN Server is installed on the system, some of the files may be protected by a facility in the LAN server, called ACL (Access Control List). The Service Pak XR06055 has a built-in facility to check for this. If during the installation, the Service Pak XR06055 finds a directory or file with ACL against it, the Service Pak will display a message that it has detected ACL and will stop the installation until all the ACLs are manually removed.
- When the installation has completed your system will automatically be configured for VGA, even if you have an SVGA adapter installed. To configure your system for SVGA, follow the the procedures outlined in 3.6, "Configuring SVGA Display Drivers" on page 35. Do not try to install the SVGA drivers manually as OS/2 sets up and configures a number of new files and they must be configured correctly by the operating system.
- After the Service Pak XR06055 has been installed, Dynamic Data Exchange (DDE) will be started automatically whenever a public session is open. The

Data Update object in the productivity folder can be shredded as it is no longer required.

A.6.2 Installing the Service Pak XR06055

There are basically two ways to install the Service Pak XR06055:

- The first, and quickest, way to install the Service Pak XR06055 is by inserting the first Service Pak XR06055 diskette into drive A: and then booting the system unit. Installing the Service Pak XR06055 in this manner will apply the changes to all the partitions where it finds the OS/2 2.0 operating system. If you have set up your system so that the operating system is on one partition and the applications are on another, the Service Pak XR06055 will only update the partition that the operating system is on.

  If you have an AOX card installed on your system it is recommended that you use this method of installing the Service Pak.

- The second method is also a fairly simple procedure, and is performed after the system has been started. To install using this method:

  1. Insert the first Service Pak XR06055 diskette into drive A: and then either enter the command **A:\SERVICE** from any OS/2 command prompt, or double click on the **Drive A** icon, and then double click on the **Service** icon.
  2. Click on **OK** and the Corrective Service Facility window will be displayed, as in Figure 196.

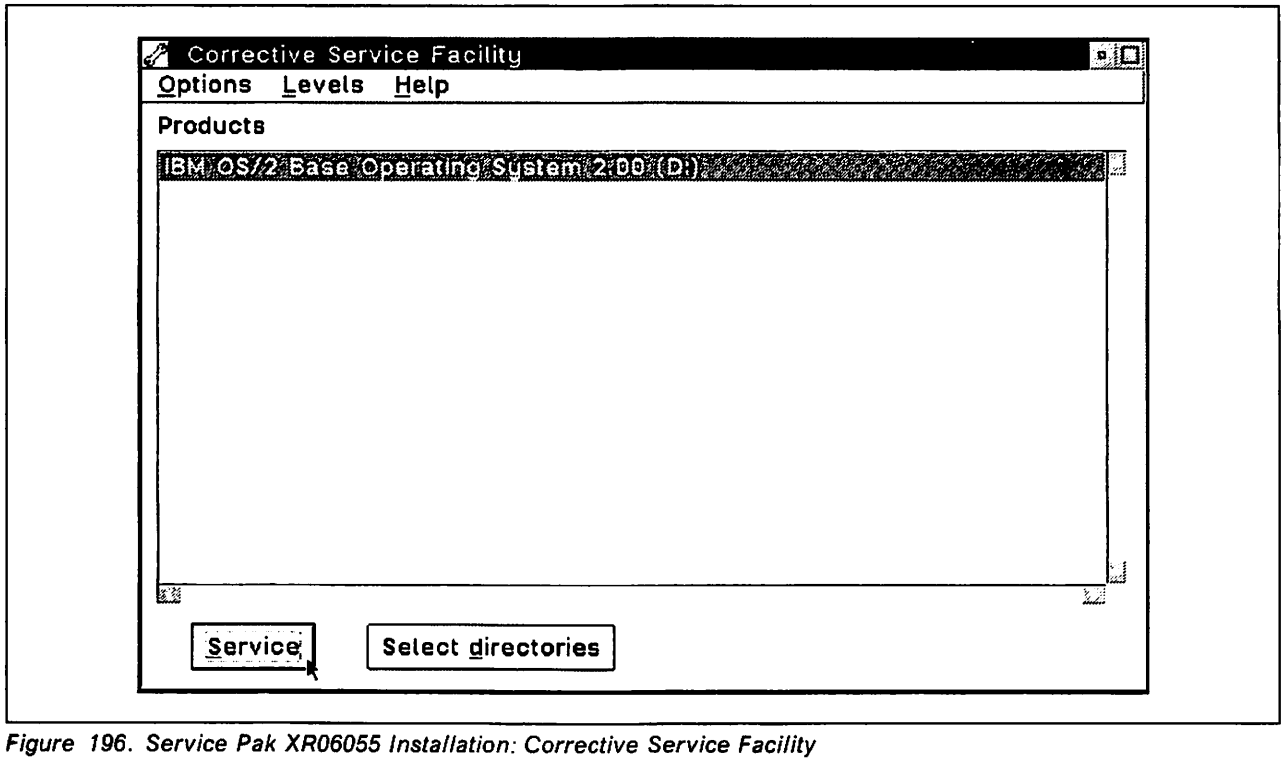


Figure 196. Service Pak XR06055 Installation: Corrective Service Facility

The Service Pak XR06055 program will have preselected the drives that should be updated. We recommend that you do not de-select any of the drives or directories that have been preselected. If you do not apply the Service Pak XR06055 updates to all directories, your system may not work when the installation completes. The reason for this is because an

updated file in one directory may be dependent on a file being updated in another directory.

3. Click on **Service** to accept the preselected directories. The Initiating Service window will be displayed as in Figure 197.

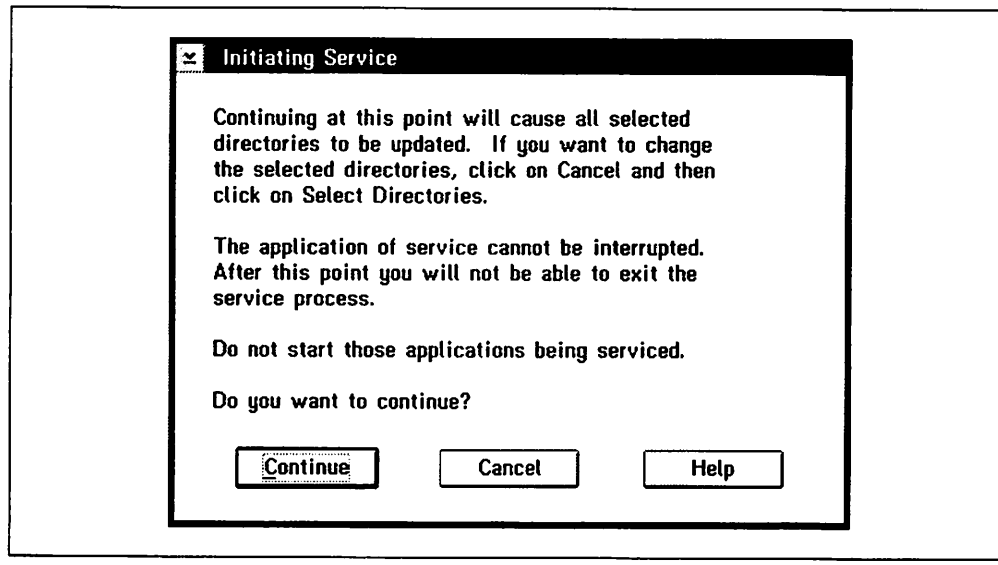


Figure 197. Service Pak XR06055 Installation: Initiating Service

4. Click on **Continue** and a Reboot system message will be displayed advising you to restart the system, as in Figure 198.

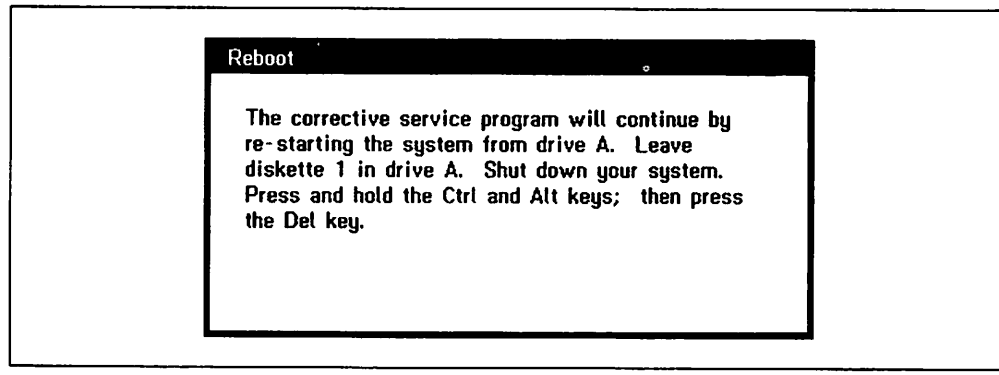


Figure 198. Service Pak XR06055 Installation: Reboot Message

5. Leave the first Service Pak XR06055 diskette in drive A: and press and hold the **Control, Alt and Delete** keys to restart your system.

   The Service Pak XR06055 Installation program will then ask you to insert the rest of the Service Pak XR06055 diskettes. Continue to do so until all the diskettes have been installed.

6. When the installation has been completed, remove any diskette from drive A: and restart your system by pressing and holding the **Control, Alt and Delete** keys.

A.6.3 Updating Printer Drivers from the Service Pak

The OS/2 Service Pak provides updates to some of the printer drivers that shipped with OS/2 2.0. These printer drivers are updated when the Service Pak is applied to your system. As this will only update the printer drivers that you have installed on your system, and the Service Pak does not contain separate Printer Driver Diskettes, you would have to reapply the Service Pak every time you installed a new printer driver from the OS/2 2.0 Printer Driver Diskettes. We recommend the following procedure to create a set of updated Printer Driver Diskettes:

1. Create a directory on your hard drive on the install drive for each Printer Driver Diskette. We used PD1, PD2, PD3 and PD4.
2. **XCOPY** each OS/2 2.0 Printer Driver Diskette into its respective directory.
3. Insert the first Service Pak diskette in drive A.
4. Type **A:SERVICE** at an OS/2 command prompt.
5. Click on the option to **Select Directories**.
6. Then do one of the following:
   a. If you have already applied the Service Pak or do not want to and only want a set of updated printer drivers then deselect all of the highlighted directories and select PD1, PD2, PD3 and PD4 which should be at the bottom of the list.
   b. If you want to apply the Service Pak *and* update the printer drivers then scroll down the list of directories until you see PD1, PD2, PD3 and PD4. Highlight these by clicking on each one.
7. Click on **Service**.
8. Insert diskettes as prompted.
9. After the Service Pak has been successfully applied you can create a set of updated Printer Driver Diskettes by using **XCOPY** from the hard drive directories to diskettes in drive A.

A.7 Summary Contents of OS/2 2.00.1, Service Pak XR06055, and OS/2 2.1

Table 31. Installation and Configuration Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
Installation from CD-ROM			Yes
Preloaded onto IBM and PCM systems	Yes		Yes
Enhancements to Selective Install program			Yes
Display Driver Install program	Yes		Yes
Warning on ACL Protection before Installation		Yes	Yes
Loadable ABIOS Installation	Backup diskettes	Yes	Yes

Table 32. Hardware Support Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
Additional SCSI Adapter support			Yes
Additional CD-ROM support			Yes
Loadable ABIOS support	Yes	Yes	Yes
PS/2 Server 195 and 295 support			Yes
Brazilian Keyboard support	Backup diskettes	Yes	Yes
Support for Enhanced 2.88MB Diskette Drive	Soft eject	Soft eject	Yes
Support for 3.5" Enhanced Rewritable Optical Drive			Yes
Pentium exploitation			Yes

Table 33. Control Program Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
Page tuning performance enhancements			Yes
XCOPY Enhancements	Yes	Yes	Yes
OS2VER File			Yes
Format Utility Enhanced for P-ROM Optical Disks			Yes

Table 34. MVDM Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
Dual-thread MVDM support	Yes	Yes	Yes
DOS_AUTOEXEC setting			Yes
DPMI 1.0 Subset support			Yes
PC Support/400 (V2R3) support			Yes
MSCDEX support			Yes

Table 35. WIN-OS/2 3.1 Support and Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
WIN-OS/2 3.1 Standard Mode support			Yes
WIN-OS/2 3.1 Enhanced Compatibility Mode support			Yes
WIN-OS/2 3.1 Performance Enhancements			Yes
Seamless WIN-OS/2 display drivers for XGA, SVGA (Tseng, Headland, Western Digital, Trident, ATI, Cirrus, and IBM Speedway), 8514 and VGA			Yes
Support for Windows 3.1 printer drivers			Yes
Improved Clipboard and DDE support			Yes
Ability to start a DOS or OS/2 application from a Windows application			Yes
Inclusion of Windows 3.1 File Manager and Selected Applets			Yes
Improved WIN-OS/2 Setup and Configuration			Yes
Improved OLE Support in WIN-OS/2 3.1			Yes
Truetype Fonts in WIN-OS/2 3.1			Yes
Multimedia support for audio in WIN-OS/2 3.1			Yes

Table 36. Presentation Manager Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
32-bit PM Graphics Engine	Yes	Yes	Yes
32-bit Seamless PM VGA display driver	Yes	Yes	Yes
32-bit Seamless PM 8514 display driver			Yes
32-bit Seamless PM SVGA IBM Speedway 256-color display driver	Yes		
32-bit Seamless PM SVGA Tseng 256-color display driver	Yes	Yes	
32-bit Seamless PM SVGA Combined 256-color display driver (Tseng, Headland, Western Digital, Trident, ATI, Cirrus, and IBM Speedway)			Yes
32-bit Seamless PM XGA display driver	Yes	Yes	Yes
ISO Font Support	Yes	Yes	Yes
Palette Manager for XGA, XGA-2, SVGA, 8514/A	Yes	Yes	Yes
XGA-2 DMQS Override	Yes	Yes	Yes

Table 37. Workplace Shell Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
Improved INI file handling			Yes
Workplace Shell Visual enhancements			Yes
Settings Notebook Drag/Drop enhancements			Yes
Auto-lockup on System Startup			Yes

Table 38. Print Subsystem Enhancements

Enhancement	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
Enhanced printer installation	Yes	Yes	Yes
Print Spooler enhancements			Yes
Additional and Enhanced PM printer drivers	Some	Some	Yes

Table 39. Enhanced Support for Laptops and Notebooks			
Enhancement	**OS/2 2.00.1**	**Service Pak XR06055**	**OS/2 2.1**
Advanced Power Management support			Yes
PCMCIA support			Yes
Trackpoint II support	Yes	Yes	Yes
Large Cursor on VGA LCD Displays	Yes	Yes	Yes

Table 40. Multimedia Support			
Enhancement	**OS/2 2.00.1**	**Service Pak XR06055**	**OS/2 2.1**
Inclusion of MMPM/2 1.1 with OS/2 2.1			Yes
Software Motion Video			Yes
Multimedia device drivers			Yes
MMPM/2 applets			Yes
MMPM/2 utilities			Yes
MMPM/2 installation			Yes
Media Control Interface subsystem			Yes
Stream Programming Interface subsystem			Yes
Multimedia I/O Services subsystem			Yes
Additional multimedia controls			Yes
Applications for MMPM/2			Yes

Appendix B. OS/2 2.x Compatible PCM Systems

OS/2 2.1 has been tested in conjunction with many PC systems from a wide range of PC Manufacturers (PCMs), by IBM at its OEM test laboratories in Boca Raton, Florida and Basingstoke, England, and also via two widespread beta tests of OS/2 2.1.

The current list of PCM hardware supported by OS/2 2.1 is included in the **PCMTABLE** package, available on CompuServe, and to IBM internal users on OS2TOOLS.

This list is maintained by IBM's Worldwide Industry Hardware Support group in the Personal Systems Programming laboratory at Boca Raton in Florida.

The contents of this hardware support list, as of April 27th 1993, has been reproduced in this appendix.

If a PC system is not in this appendix, please check CompuServe for a more recent version of the list, or contact the hardware manufacturer.

B.1 Introduction to Compatibility Tables for PCM Systems

The IBM OS/2 2.x operating system supports all Intel 386SX and higher (or compatible) based Personal Computer Manufacturer (PCM) systems in the marketplace. The following table depicts those PCM systems that have passed compatibility testing with the IBM OS/2 2.x product. IBM compatibility testing verifies (18) key functions of the OS/2 2.x operating system. These test results are based on selected PCM model configurations tested.

The following IBM OS/2 products have been tested on a significant number of PCM systems listed in the table below. Based on these tests, IBM has determined that these products are considered compatible with all the PCM systems listed.

- OS/2 Extended Services Version 1.0
- Extended Services CID Utility
- Distributed Database Connection Services/2 (DDCS/2)
- OS/2 LAN Server Version 2.0
- OS/2 LAN Server Version 3.0
- OS/2 LAN Enabler Version 2.0
- LAN NetView Start Version 1.0
- NTS/2 Version 1.0
- System Performance Monitor/2 Version 2.0
- NetWare from IBM products

Note

This table is available electronically in the USA on the IBM National Support Center Bulletin Board System (data connection: 1-404-835-6600), and on CompuServe (IBM OS/2 Support Forum \ Library \ IBMFiles \ PCMTABLE).

B.2 Personal Computer Manufacturer's Systems

Table 41 (Page 1 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
ACER 1125E	YES	YES	YES
ACER AcerFrame 300 486DX/33	YES	YES	YES
ACER AcerFrame 1000	YES	YES	YES
ACER AcerFrame 3000 MP/33	YES	YES	YES
ACER AcerMate 386/33	YES	YES	YES
ACER AcerMate 386SX/20N	YES	YES	YES
ACER AcerPower 386SX	YES	YES	YES
ACER AcerPower 425s	YES	YES	YES
ACER AcerPower 433	YES	YES	YES
ACER AcerPower 433e	YES	YES	YES
ACER AcerPower 486SX	YES	YES	YES
ACER AcerPower 500	YES	YES	YES
Advanced Network Communication 80486VESA 33 MHZ	YES	YES	YES
Advanced Network Communication 80486VESA 50 MHZ	YES	YES	YES
AEG OLYMPIA Olystar 70V	YES	YES	YES
AEG OLYMPIA Olystar 80S	YES	YES	YES
ALR BusinessSTATION 386DX Model 1	YES	YES	YES
ALR BusinessSTATION 486SX/20 Model 101	YES	YES	YES
ALR BusinessVEISA 386/33 Model 1	YES	YES	YES
ALR BusinessVEISA 486/33 Model 101	YES	YES	YES
ALR MPS Modular 386/33 Model 1	YES	YES	YES
ALR MPS Modular 486/33 Model 101	YES	YES	YES
ALR PowerFlex 20SX Model 80	YES	YES	YES
ALR PowerFlex FLYER Model 60	YES	YES	YES
ALR PowerPro 33/486 Model SMP 128/150	YES	YES	YES
AMAX PC/486	YES	YES	YES
AMBRA Enterprise Slim Profile 386SX/33	YES	YES	YES
AMBRA Enterprise Slim Profile 486SX/25	YES	YES	YES
AMBRA Enterprise Full Desktop 486SX/25	YES	YES	YES
AMBRA Enterprise Full Desktop 486DX/33	YES	YES	YES
AMBRA Enterprise MAX 486DX/33	YES	YES	YES
AMBRA Enterprise MAX 486DX/50	YES	YES	YES
AMBRA Enterprise MAX 486DX2/50	YES	YES	YES
AMBRA Enterprise MAX 486DX2/66	YES	YES	YES
AMBRA Hurdla 386SX/33	YES	YES	YES
AMBRA Hurdla 486SX/25	YES	YES	YES
AMBRA Hurdla 486DX/33	YES	YES	YES
AMBRA Hurdla 486DX/50	YES	YES	YES
AMBRA Hurdla 486DX2/50	YES	YES	YES
AMBRA Hurdla 486DX2/66	YES	YES	YES
AMBRA Sprinta 386SX/33	YES	YES	YES
AMBRA Sprinta 486SX/25	YES	YES	YES
AMD AmSCSIXPRS-94/M2-SCSI	YES	YES	YES
AMI EZ-Flex	YES	YES	YES
AMSTRAD PC3386 SX	YES	YES	YES
AOX Micromaster 386 in PS2 Model 60	YES	YES	YES

Table 41 (Page 2 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
AOX Micromaster 486 in PS2 Model 55	YES	YES	YES
APD Series/40	YES	YES	YES
APRICOT LS 386SX-20	YES	YES	YES
APRICOT LS PRO 386SX	YES	YES	YES
APRICOT Qi 386-25	YES	YES	YES
APRICOT Qi 486-25	YES	YES	YES
ARCHE Legacy 486	YES	YES	YES
ARES TP 386/33 SONIC	YES	YES	YES
ARES TP 486/33 SONIC	YES	YES	YES
ASEM DS 486/33	YES	YES	YES
ASI COMPUTER Easyline 486 SX 25	YES	YES	YES
ASI COMPUTER Megaline 486/33	YES	YES	YES
AST Bravo 3/25s	YES	YES	YES
AST Bravo 386SX/20	YES	YES	YES
AST Bravo 486/25 83V	YES	YES	YES
AST Power Exec 4/25 SL	YES	YES	YES
AST Power Premium 4/33	YES	YES	YES
AST Premium 386/33	YES	YES	YES
AST Premium 486/33E	YES	YES	YES
AST Premium 486/33TE	YES	YES	YES
AST Premium II 386SX/20	YES	YES	YES
AST Premium II 386/33	YES	YES	YES
AST Premium II 486SX/20	YES	YES	YES
AST Premium EXEC 386SX/20	YES	N/A	N/A
AST Premium EXEC 386SX/25 Color	YES	N/A	N/A
AST Premium SE 4/33	YES	YES	YES
AST Premium SE 4/50	YES	YES	YES
AST Premmia 4/66d	YES	YES	YES
AST SE 4/66	YES	YES	YES
ATOMSTYLE APC 386-33	YES	YES	YES
AT&T Safari	YES	N/A	N/A
AT&T StarServer S	YES	YES	YES
AT&T 6386/25 WGS	YES	YES	YES
AT&T 6386SX/EL20 WGS	YES	YES	YES
CAS 386SX-20	YES	YES	YES
C & C COMPUTERS MT 450/256	YES	YES	YES
CHEM CORP 386DX-40 Opti	YES	YES	YES
CHEM CORP 486SX-33	YES	YES	YES
CLUB AMERICA FALCON 433	YES	YES	YES
COMMODORE 386-33C	YES	YES	YES
COMMODORE 386SX-20	YES	YES	YES
COMMODORE 486-33C	YES	YES	YES
COMPAQ 386/20	YES	YES	YES
COMPAQ Contura 4/25	YES	YES	YES
COMPAQ Contura 4/25c	YES	YES	YES
COMPAQ LTE-386s/20	YES	YES	YES
COMPAQ DeskPro 386/25e	YES	YES	YES
COMPAQ DeskPro 386/25M	YES	YES	YES

Table 41 (Page 3 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
COMPAQ DeskPro 386/33L	YES	YES	YES
COMPAQ DeskPro 386/N	YES	YES	YES
COMPAQ DeskPro 386/s	YES	YES	YES
COMPAQ DeskPro 386S/20	YES	YES	YES
COMPAQ DeskPro 486/16M	YES	YES	YES
COMPAQ DeskPro 486/50L	YES	YES	YES
COMPAQ DeskPro 486/66i	YES	YES	YES
COMPAQ ProSignia	YES	YES	YES
COMPAQ SystemPro 386	YES	YES	YES
COMPAQ SystemPro 486	YES	YES	YES
CompuAdd 320sc	YES	YES	YES
CompuAdd 325	YES	YES	YES
CompuAdd 333T	YES	YES	YES
CompuAdd 433E	YES	YES	YES
COMPUTER PLUS Perpetual Systems 386-SX	YES	YES	YES
COMPUTER PLUS Perpetual Systems 386-DX	YES	YES	YES
COMPUTER PLUS Perpetual Systems 386-DX/LB	YES	YES	YES
COMPUTER PLUS Perpetual Systems 486-SX	YES	YES	YES
COMPUTER PLUS Perpetual Systems 486-DX	YES	YES	YES
COMPUTER PLUS Perpetual Systems 486-LB	YES	YES	YES
COMPUTER PLUS Perpetual Systems 486-EISA	YES	YES	YES
COPAM PC 486B/33	YES	YES	YES
CRITIKON 8600	YES	YES	YES
CUBE COMPUTER 340 ATX	YES	YES	YES
CUBE COMPUTER 433 ATX	YES	YES	YES
CUBE COMPUTER 450 ATX	YES	YES	YES
CUMULUS 486SX/20	YES	YES	YES
CUMULUS GLC 386SX/20	YES	YES	YES
CUMULUS GLC 386DX/25	YES	YES	YES
CUMULUS GLC 386DX/33	YES	YES	YES
CUMULUS GLC 486DX/33	YES	YES	YES
CUMULUS WORKBOX 16	YES	YES	YES
CUMULUS WORKBOX 20	YES	YES	YES
DALY COMPUTER Daly386-40	YES	YES	YES
DALY COMPUTER Daly486-33	YES	YES	YES
DALY COMPUTER Daly486E-50	YES	YES	YES
DDI Dynex 386-25	YES	YES	YES
DDI Dynex Professional 486/50	YES	YES	YES
DEC DECpc 316sx	YES	YES	YES
DEC DECpc 320	YES	YES	YES
DEC DECpc 325	YES	YES	YES
DEC DECpc 325sx LP	YES	YES	YES
DEC DECpc 333	YES	YES	YES
DEC DECpc 333sx LP	YES	YES	YES
DEC DECpc 340dx LP	YES	YES	YES
DEC DECpc 420sx	YES	YES	YES
DEC DECpc 425	YES	YES	YES
DEC DECpc 425i	YES	YES	YES

Table 41 (Page 4 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
DEC DECpc 425i/dx2	YES	YES	YES
DEC DECpc 425sx	YES	YES	YES
DEC DECpc 425 ST	YES	YES	YES
DEC DECpc 433dx LP	YES	YES	YES
DEC DECpc 433dx MP	YES	YES	YES
DEC DECpc 433 ST	YES	YES	YES
DEC DECpc 433 T	YES	YES	YES
DEC DECpc 433 Workstation	YES	YES	YES
DEC DECpc 450d2 LP	YES	YES	YES
DEC DECpc 450d2 MP	YES	YES	YES
DEC DECpc 450 ST	YES	YES	YES
DEC DECpc 452 ST	YES	YES	YES
DEC DECpc 466D2 LP	YES	YES	YES
DEC DECpc 466D2 MT	YES	YES	YES
DEC DECpc 466 ST	YES	YES	YES
DEC DECstation 320	YES	YES	YES
DEC DECstation 320+	YES	YES	YES
DEC DECstation 333c	YES	YES	YES
DELL 320N+	YES	N/A	N/A
DELL 320SLi	YES	YES	YES
DELL 320SX	YES	YES	YES
DELL 325P	YES	YES	YES
DELL 333D	YES	YES	YES
DELL 433DE	YES	YES	YES
DELL 433/L	YES	YES	YES
DELL 433s/ME	YES	YES	YES
DELL 450/M	YES	YES	YES
DELL 450SE	YES	YES	YES
DELL 466/T	YES	YES	YES
DELL 486D/25	YES	YES	YES
DELL 486P/33	YES	YES	YES
DTK COMPUTER Grafika - 4A	YES	YES	YES
DTK COMPUTER Grafika - 4C	YES	YES	YES
DTK COMPUTER Grafika - 4D	YES	YES	YES
DTK COMPUTER Grafika - 4E	YES	YES	YES
DTK COMPUTER Grafika - 4F	YES	YES	YES
DTK COMPUTER Grafika - 4G	YES	YES	YES
DTK COMPUTER Grafika - 4H	YES	YES	YES
DTK COMPUTER Grafika - 4I	YES	YES	YES
DTK COMPUTER Grafika - 4J	YES	YES	YES
EPSON T0341U	YES	YES	YES
EPSON T2331U	YES	YES	YES
EPSON EISA Series T2331C	YES	YES	YES
EPSON EISA Series T2331U	YES	YES	YES
EPSON EISA Series 4s/25Te	YES	YES	YES
EPSON EISA Series 4/33Te	YES	YES	YES
EPSON EISA Series 4/50De	YES	YES	YES
EPSON EISA Series 4/50Te	YES	YES	YES

Table 41 (Page 5 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
EPSON Equity 320SX Plus	YES	YES	YES
EPSON Equity 386/33 Plus	YES	YES	YES
EPSON Equity 486DX2/50 Plus	YES	YES	YES
EPSON ExpressStation 4s/25	YES	YES	YES
EPSON ExpressStation 4/33	YES	YES	YES
EVEREX 386/33 EX0-2804D-000L	YES	YES	YES
EVEREX 486SX/20 EX0-2904A-010L	YES	YES	YES
EVEREX 486SX/25	YES	YES	YES
EVEREX 486/33 EX0-2904D-B1	YES	YES	YES
EVEREX 486/33e EX0-2904D-01EL	YES	YES	YES
EVEREX 486/33e EX0-2908D-01EC	YES	YES	YES
EVEREX 486DX2	YES	YES	YES
EVEREX 486DX2/50 EX0-2804P-E1ET	YES	YES	YES
EVEREX DX2/50	YES	YES	YES
EVEREX Step MegaCube DX/50e	YES	YES	YES
EVEREX Tempo 386/33 EX0-4408M-00HL	YES	YES	YES
EVEREX Tempo 386/33c EX0-4404M-00HS	YES	YES	YES
EVEREX Tempo 486/20 EX0-4504J-00HL	YES	YES	YES
EVEREX Tempo 486SX/20c EX0-4504J-00HS	YES	YES	YES
EVEREX Tempo 486/33 EX0-4608M-00HL	YES	YES	YES
EVEREX Tempo 486/33c EX0-4608M-00HS	YES	YES	YES
EVEREX Tempo 486DX2/50 M	YES	YES	YES
EVEREX Tempo Standard 386/25	YES	YES	YES
EVEREX Tempo Standard 386SX/16	YES	YES	YES
EVEREX Tempo Standard 386SX/20	YES	YES	YES
EVEREX Tempo Standard 386SX/25	YES	YES	YES
GATEWAY 2000 386SX/20C	YES	YES	YES
GATEWAY 2000 386/33C	YES	YES	YES
GATEWAY 2000 486/33C	YES	YES	YES
GATEWAY 2000 486/33E	YES	YES	YES
GRiD 386is-25	YES	YES	YES
GRiD 386sx-MFP20	YES	YES	YES
GRiD 4020SX	YES	YES	YES
GRiD 4025LS	YES	YES	YES
GRiD 4025LX	YES	YES	YES
GRiD GRiDCASE 1550sx	YES	N/A	N/A
GRiD APT/425se	YES	YES	YES
GRiD APT/450e	YES	YES	YES
GRiD MFP/420s	YES	YES	YES
GRiD MFP/425s	YES	YES	YES
GRiD MFP 425s+	YES	YES	YES
GRiD MFP 433+	YES	YES	YES
GRiD MFP 433s+	YES	YES	YES
GRiD MFP/450	YES	YES	YES
GRiD MFP 450+	YES	YES	YES
GRiD MFP 466+	YES	YES	YES
GRiD MFP/540	YES	YES	YES
G2 COMPUTER SYSTEMS S1-SX	YES	YES	YES

Table 41 (Page 6 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
HERTZ 486/33	YES	YES	YES
HEWITT RAND CORPORATION 313/318	YES	YES	YES
HEWLETT PACKARD QS/20	YES	YES	YES
HEWLETT PACKARD TestBook	YES	YES	YES
HEWLETT PACKARD Vectra 386/16N	YES	YES	YES
HEWLETT PACKARD Vectra 386/25	YES	YES	YES
HEWLETT PACKARD Vectra 486/25T	YES	YES	YES
HEWLETT PACKARD Vectra 486s/20	YES	YES	YES
HEWLETT PACKARD Vectra RS/25C	YES	YES	YES
HEWLETT PACKARD Vectra 486/33T	YES	YES	YES
HEWLETT PACKARD Vectra 486/50U	YES	YES	YES
HYPERTEC 386SX/25 - PS/2 Mod 50Z	YES	YES	YES
HYPERTEC 386SX/33 - PS/2 Mod 50Z	YES	YES	YES
HYPERTEC 486/33 - PS/2 Mod 80	YES	YES	YES
HYUNDAI ELECTRONICS AMERICA 425SP	YES	YES	YES
HYUNDAI ELECTRONICS AMERICA 433DP	YES	YES	YES
HYUNDAI ELECTRONICS AMERICA 433SP	YES	YES	YES
HYUNDAI ELECTRONICS AMERICA 450D2P	YES	YES	YES
IBM PS/1 2133 386SX/25	YES	YES	YES
IBM PS/1 2133 486SX/20	YES	YES	YES
IBM PS/1 2133 486DX/20	YES	YES	YES
IBM PS/1 2133 486DX/40	YES	YES	YES
IBM PS/1 2155 386SX/25	YES	YES	YES
IBM PS/1 2155 486SX/20	YES	YES	YES
IBM PS/1 2155 486DX/20	YES	YES	YES
IBM PS/1 2155 486DX/33	YES	YES	YES
IBM PS/1 2155 486DX/40	YES	YES	YES
IBM PS/2 8540 386SX/20	YES	YES	YES
IBM PS/2 8543 386SX/20	YES	YES	YES
IBM PS/2 8555 386SX/16	YES	YES	YES
IBM PS/2 8556 386SX/20	YES	YES	YES
IBM PS/2 8556 386SLC/20	YES	YES	YES
IBM PS/2 8557 386SX/20	YES	YES	YES
IBM PS/2 8557 386SLC/20	YES	YES	YES
IBM PS/2 8565 386SX/16	YES	YES	YES
IBM PS/2 8570 386DX/16-25	YES	YES	YES
IBM PS/2 8570 486DX/25	YES	YES	YES
IBM PS/2 8573 386DX/16-20	YES	YES	YES
IBM PS/2 8575 486DX/33	YES	YES	YES
IBM PS/2 8580 386DX/20-25	YES	YES	YES
IBM PS/2 8590 486SX/25	YES	YES	YES
IBM PS/2 8590 486DX/25-33	YES	YES	YES
IBM PS/2 8590 486DX2/50-66	YES	YES	YES
IBM PS/2 8595 486SX/25	YES	YES	YES
IBM PS/2 8595 486DX/25-33	YES	YES	YES
IBM PS/2 8595 486DX2/50-66	YES	YES	YES
IBM PS/2 9556 486SLC2	YES	YES	YES
IBM PS/2 9557 486SLC2	YES	YES	YES

Table 41 (Page 7 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
IBM PS/2 9576 486SX	YES	YES	YES
IBM PS/2 9576 486DX2	YES	YES	YES
IBM PS/2 9577 486SX	YES	YES	YES
IBM PS/2 9577 486DX2	YES	YES	YES
IBM PS/2 9585 486SX	YES	YES	YES
IBM PS/2 9585 486DX2	YES	YES	YES
IBM PS/2 9595 486DX	YES	YES	YES
IBM PS/2 9595 486DX2	YES	YES	YES
IBM PS/2 295 486DX	YES	YES	YES
IBM PS/ThinkPad N45SL 386SL	YES	YES	YES
IBM PS/ThinkPad N51SX 386SX	YES	YES	YES
IBM PS/ThinkPad N51SLC 386SLC	YES	YES	YES
IBM PS/ThinkPad CL57SX 386SX	YES	YES	YES
IBM PS/ThinkPad 300 386SL	YES	YES	YES
IBM PS/ThinkPad 700 486SLC	YES	YES	YES
IBM PS/ThinkPad 700C 486SLC	YES	YES	YES
IBM PS/ValuePoint 325T	YES	YES	YES
IBM PS/ValuePoint 425SX	YES	YES	YES
IBM PS/ValuePoint 433DX	YES	YES	YES
IBM PS/ValuePoint 466DX2	YES	YES	YES
ICL Personal Computer CL386S	YES	YES	YES
ICL Personal Computer CL386s/25	YES	YES	YES
ICL Personal Computer CL486	YES	YES	YES
ICL Personal Computer CL486s/25	YES	YES	YES
ICL Personal Computer CS386s	YES	YES	YES
ICL Personal Computer CS386s/25	YES	YES	YES
ICL Personal Computer CX386s	YES	YES	YES
ICL Personal Computer CX386s/25	YES	YES	YES
ICL Personal Computer CX486	YES	YES	YES
ICL Personal Computer CX486s/25	YES	YES	YES
ICL Personal Computer CXe486	YES	YES	YES
ICL Personal Computer CXe486s	YES	YES	YES
ICL Personal Computer CXe486/66	YES	YES	YES
ICL Personal Computer FX486	YES	YES	YES
ICL Personal Computer FX486s	YES	YES	YES
ICL Personal Computer FX486/66	YES	YES	YES
ICL Personal Computer NB386L	YES	YES	YES
ICL Personal Computer NB386LC	YES	YES	YES
INACOM 386/33	YES	YES	YES
INTEL DT386-33H	YES	YES	YES
INTEL iSBC 486/DX6600V	YES	YES	YES
INTEL L486 Series	YES	YES	YES
INTEL LP486SX25E	YES	YES	YES
INTEL Professional GX	YES	YES	YES
INTEL SnapIn/386 - IBM PC/AT	YES	YES	YES
INTEL SnapIn/386 - PS2 Model 50Z	YES	YES	YES
INTEL SnapIn/386 - PS2 Model 60	YES	YES	YES
INTEL Xpress Desktop 50mhz i486DX Platform	YES	YES	YES

Table 41 (Page 8 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
INTERCOMP Digit 486SLC-25	YES	YES	YES
INTERCOMP Master 486DX-50	YES	YES	YES
INTERCOMP Master 486DX2-66	YES	YES	YES
INTERCOMP Planet 486DX-50 EISA	YES	YES	YES
INTERCOMP Target 486SLC-25	YES	YES	YES
INTERCOMP Target 486DX-33	YES	YES	YES
INVESTRONICA WS-900 SX	YES	YES	YES
INVESTRONICA WS-600 CACHE	YES	YES	YES
ITAUTEC IS 486 D33E (IDE)	YES	YES	YES
ITAUTEC IS 486 D33E (SCSI)	YES	YES	YES
JEN ELETIRONICA S.R.L. PR 466E/256	YES	YES	YES
JEN ELETIRONICA S.R.L. ST 386SX33/32	YES	YES	YES
JEN ELETIRONICA S.R.L. ST 486/66	YES	YES	YES
KINGSTON SX/Now! 50Z	YES	YES	YES
KINGSTON SX/Now! 60	YES	YES	YES
KINGSTON SX/Now SX25/GAM-PS/2 Mod 50	YES	YES	YES
KINGSTON SX/Now SX25/LCM-PS/2 Mod 50Z	YES	YES	YES
KINGSTON SX/Now SX25/GAM-PS/2 Mod 60	YES	YES	YES
KINGSTON SX/Now SX33/LCM-PS/2 Mod 50Z	YES	YES	YES
KONTRON ELEKTRONIK IP Lite Color 486	YES	YES	YES
KREX COMPUTERS Krex 486	YES	YES	YES
LEADING EDGE MT33	YES	YES	YES
LEADING EDGE D4/SX20	YES	YES	YES
LOCLAND 486-SX Convertible	YES	YES	YES
LOCLAND 486-33 Convertible	YES	YES	YES
LOCLAND 486-50 Convertible	YES	YES	YES
LOCLAND 486-50 EISA Converible	YES	YES	YES
MEMOREX TELEX Model 8090-50	YES	YES	YES
MEMOREX TELEX Model 8257	YES	YES	YES
MEMOREX TELEX Model 8267	YES	YES	YES
MEMOREX TELEX Model 8280	YES	YES	YES
MEMOREX TELEX Model 8290	YES	YES	YES
MIND COMPUTER PRODUCTS 486SX-25	YES	YES	YES
MIND COMPUTER PRODUCTS 486DX-33	YES	YES	YES
MIND COMPUTER PRODUCTS 486DX-33 HW-HDC-GEN-013	YES	YES	YES
MINI-MICRO 386	YES	YES	YES
MINI-MICRO 486	YES	YES	YES
MITAC MiStation 3052E	YES	YES	YES
MITAC Personal Computer 3060F	YES	YES	YES
MONYDATA 386 SX	YES	YES	YES
MONYDATA Modula 60	YES	YES	YES
NCR System 3220/C3220 Model 0100	YES	YES	YES
NCR System 3320/C3421 Model 2014	YES	YES	YES
NCR System 3335/C3432 Model 1000	YES	YES	YES
NCR System 3335/C3432 Model 5000	YES	YES	YES
NCR System 3340/C3314 Model 1000	YES	YES	YES
NCR System 3345/C3433 Model 2000	YES	YES	YES
NCR System 3345/C3433 Model 5000	YES	YES	YES

Table 41 (Page 9 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
NCR System 3350/C3350 Model 1000	YES	YES	YES
NCR System 3350/C3350 Model 4000	YES	YES	YES
NCR System 3355/C3355 Model 1100	YES	YES	YES
NCR System 3355/C3355 Model 4100	YES	YES	YES
NCR System 3420/C3420	YES	YES	YES
NCR System 3445/C3434 Model 3000	YES	YES	YES
NCR System 3445/C3434 Model 5000	YES	YES	YES
NCR System 3447/C3437 Model 3000	YES	YES	YES
NEC Image 425	YES	YES	YES
NEC Image 433	YES	YES	YES
NEC Image 466	YES	YES	YES
NEC PowerMate SX/20i	YES	YES	YES
NEC PowerMate SX/20vi	YES	YES	YES
NEC PowerMate 386/33E	YES	YES	YES
NEC PowerMate 386/33i	YES	YES	YES
NEC PowerMate 425	YES	YES	YES
NEC PowerMate 433	YES	YES	YES
NEC PowerMate 486SX/25i	YES	YES	YES
NEC PowerMate 486/25E	YES	YES	YES
NEC PowerMate 486/33e	YES	YES	YES
NEC PowerMate 486/33i	YES	YES	YES
NEC PowerMate 486/50e	YES	YES	YES
NEC PowerMate 486/50i	YES	YES	YES
NEC PowerMate 486/66i	YES	YES	YES
NEC PowerMate DX2/66e	YES	YES	YES
NEC Ready 325	YES	YES	YES
NEC Ready 425	YES	YES	YES
NEC Ready 433	YES	YES	YES
NEC Ready 466	YES	YES	YES
NEC UltraLite SL/20	YES	YES	YES
NIXDORF PWS M45	YES	YES	YES
NIXDORF PWS 45/50 SCSI	YES	YES	YES
NIXDORF PWS M50	YES	YES	YES
NORMEREL NS 48	YES	YES	YES
NORMEREL NS 78	YES	YES	YES
NORTHGATE Elegance 433E	YES	YES	YES
NORTHGATE Elegance ZXP	YES	YES	YES
NOTESTAR NP-925	YES	YES	YES
NOTESTAR NP-933	YES	YES	YES
OCCIDENTAL SYSTEMS Executive 486DX-33	YES	YES	YES
OCCIDENTAL SYSTEMS Upgradeable 486DX-33	YES	YES	YES
OLIVETTI LSX5010	YES	YES	YES
OLIVETTI LSX5015	YES	YES	YES
OLIVETTI LSX5020	YES	YES	YES
OLIVETTI LSX5025	YES	YES	YES
OLIVETTI M300-08	YES	YES	YES
OLIVETTI M300-10	YES	YES	YES
OLIVETTI M300-15	YES	YES	YES

Table 41 (Page 10 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
OLIVETTI M300-25	YES	YES	YES
OLIVETTI M300-30	YES	YES	YES
OLIVETTI M300-30/P	YES	YES	YES
OLIVETTI M380-40	YES	YES	YES
OLIVETTI M400-10	YES	YES	YES
OLIVETTI M400-60	YES	YES	YES
OLIVETTI M480-30	YES	YES	YES
OLIVETTI P750	YES	YES	YES
PACKARD-BELL 1110 486sx/25	YES	YES	YES
PACKARD-BELL 1150 486sx/25	YES	YES	YES
PACKARD-BELL Axcel 105 486sx/25	YES	YES	YES
PACKARD-BELL Axcel 130 486sx/25	YES	YES	YES
PACKARD-BELL Axcel 2005 486sx/25	YES	YES	YES
PACKARD-BELL Axcel 2015 486sx/25	YES	YES	YES
PACKARD-BELL Force 107 486sx/25	YES	YES	YES
PACKARD-BELL Force 110 486sx/25	YES	YES	YES
PACKARD-BELL Force 117 486sx/25	YES	YES	YES
PACKARD-BELL Force 2010 486sx/25	YES	YES	YES
PACKARD-BELL Force 2020 486sx/25	YES	YES	YES
PACKARD-BELL Legend 102 Elite 486sx/25	YES	YES	YES
PACKARD-BELL Legend 102H 486sx/25	YES	YES	YES
PACKARD-BELL Legend 115 486sx/25	YES	YES	YES
PACKARD-BELL Legend 120 486sx/25	YES	YES	YES
PACKARD-BELL Legend 125 486sx/25	YES	YES	YES
PACKARD-BELL Legend 126 Elite 486sx/25	YES	YES	YES
PACKARD-BELL Legend 127 486sx/25	YES	YES	YES
PACKARD-BELL Legend 128 486sx/25	YES	YES	YES
PACKARD-BELL Legend 135 486sx/25	YES	YES	YES
PACKARD-BELL Legend 135H 486sx/25	YES	YES	YES
PACKARD-BELL Legend 140 486sx/25	YES	YES	YES
PACKARD-BELL Legend 695 Supreme 486sx/25	YES	YES	YES
PACKARD-BELL Legend 1900 486sx/25	YES	YES	YES
PACKARD-BELL Legend 1910 486sx/25	YES	YES	YES
PACKARD-BELL Legend 2000 486sx/25	YES	YES	YES
PACKARD-BELL Legend 2001 486sx/25	YES	YES	YES
PACKARD-BELL Legend 2002 Elite 486sx/25	YES	YES	YES
PACKARD-BELL Legend 2011 Supreme 486sx/25	YES	YES	YES
PACKARD-BELL Legend 2025 486sx/25	YES	YES	YES
PACKARD-BELL PackMate 2050 486sx/25	YES	YES	YES
PACKARD-BELL 486-33	YES	YES	YES
PANATEK 486DX-33	YES	YES	YES
PHILIPS P3348	YES	YES	YES
PHILIPS P3371	YES	YES	YES
PHILIPS P3448	YES	YES	YES
PHOCUS PD340-240	YES	YES	YES
PHOCUS PS333-105	YES	YES	YES
PHOCUS PST433-425	YES	YES	YES
PHOCUS PT433-670	YES	YES	YES

Table 41 (Page 11 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
PHOCUS PW325-105	YES	YES	YES
PRINTER PRODUCTS PCPOS 2000	YES	YES	YES
PROFESSIONAL COMPUTER MFG Emerald Flexi 3486	YES	YES	YES
PROFESSIONAL COMPUTER MFG Emerald Flexi-486V	YES	YES	YES
PROLOGICA ATSP 386DX	YES	YES	YES
PROLOGICA ATSP 486	YES	YES	YES
REEVES 386SX-25	YES	YES	YES
REEVES 386SX-25C	YES	YES	YES
REEVES 386-33C	YES	YES	YES
REEVES 486SX-20C	YES	YES	YES
REEVES 486-33C	YES	YES	YES
REPLY Upgrade Planar / 60-65	YES	YES	YES
REPLY Turbo Upgrade Planar / 50-50Z	YES	YES	YES
REPLY Turbo Upgrade Planar PS/2-55	YES	YES	YES
REPLY D16 3SX-160 Model 16 MCA	YES	YES	YES
REPLY D32 386-20C Model 32 MCA	YES	YES	YES
REPLY D32 386-25C Model 32 MCA	YES	YES	YES
REPLY D32 386-33C Model 32 MCA	YES	YES	YES
REPLY D32 4SX-200 Model 32 MCA	YES	YES	YES
REPLY D32 486-250 Model 32 MCA	YES	YES	YES
REPLY D32 486-330 Model 32 MCA	YES	YES	YES
REPLY D32 486-50C Model 32 MCA	YES	YES	YES
REPLY D32 486-50D Model 32 MCA	YES	YES	YES
SCANDIC PRODUCTS 1-34201i	YES	YES	YES
SCANDIC PRODUCTS 1-45201i	YES	YES	YES
SCANDIC PRODUCTS 3-43201E	YES	YES	YES
SCANDIC PRODUCTS 3-45201E	YES	YES	YES
SEC DeskMaster 386S/25	YES	YES	YES
SEC DeskMaster 486C/25	YES	YES	YES
SEC DeskMaster 486S/25N	YES	YES	YES
SEC DeskMaster 486/33E	YES	YES	YES
SEC NoteMaster 386S/25	YES	YES	YES
SEC NoteMaster 486C/25	YES	N/A	N/A
SEC NoteMaster 486S/25	YES	N/A	N/A
SIDUS SYSTEMS Formula 386-33/DT	YES	YES	YES
SIDUS SYSTEMS Formula 486sx-25/LP	YES	YES	YES
SIDUS SYSTEMS SCI925Dsx/LP	YES	YES	YES
SIEMENS NIXDORF PCD-3Bsx	YES	YES	YES
SIEMENS NIXDORF PCD-3Msx/20	YES	YES	YES
SIEMENS NIXDORF PCD-4H	YES	YES	YES
SIEMENS NIXDORF PCD-4T/33	YES	YES	YES
SIEMENS NIXDORF PCE-4C	YES	YES	YES
SIEMENS NIXDORF PCM-3Dsx/16	YES	YES	YES
SIEMENS NIXDORF PCM-4T	YES	YES	YES
SPEAR TECHNOLOGY Shuttle HOT 307-40	YES	YES	YES
STAVER COMPUTER 386SX40	YES	YES	YES
STAVER COMPUTER PC386SX-33	YES	YES	YES
STAVER COMPUTER PC386SX40-80	YES	YES	YES

Table 41 (Page 12 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
STAVER COMPUTER PC486DLC-40	YES	YES	YES
STAVER COMPUTER PC486DLC-80	YES	YES	YES
STAVER COMPUTER PC486DLC33-720	YES	YES	YES
Swan 386/33M	YES	YES	YES
Tandon OPTIoN T30000	YES	YES	YES
Tandon OPTIoN PRO T30050	YES	YES	YES
Tandon PAC II 486/33	YES	YES	YES
TANDY 425 sx	YES	YES	YES
TANDY 433 sx	YES	YES	YES
TANDY 433 DX	YES	YES	YES
TANDY 450 DX2	YES	YES	YES
TANDY 466 DX2	YES	YES	YES
TANDY 2500 sx/33	YES	YES	YES
TANDY 4020 SX/386	YES	YES	YES
TANDY 4025 LX/386	YES	YES	YES
TANDY 4825 sx	YES	YES	YES
TANDY 4833 LX/T	YES	YES	YES
TANDY 4850 EP	YES	YES	YES
TANDY 4866 LX/T	YES	YES	YES
TATUNG TCS-8160S	YES	YES	YES
TATUNG TCS-8460S	YES	YES	YES
TATUNG TCS-8800D	YES	YES	YES
TATUNG TCS-8960S	YES	YES	YES
TATUNG TCS-9600T	YES	YES	YES
Texas Instruments 386SX/20	YES	YES	YES
Texas Instruments 386SX/25	YES	YES	YES
Texas Instruments 386SXP	YES	YES	YES
Texas Instruments 386/33	YES	YES	YES
Texas Instruments 386/33P	YES	YES	YES
Texas Instruments 486/33E	YES	YES	YES
Texas Instruments 486/33TE	YES	YES	YES
Texas Instruments 486/50TE	YES	YES	YES
Texas Instruments TravelMate 3000 Series	YES	YES	YES
TEXAS MICRO SYSTEMS C-486SE/33 3014A	YES	YES	YES
THONNES DATENSYSTEME DX386-33	YES	YES	YES
THONNES DATENSYSTEME DX486-33	YES	YES	YES
TI'KO PS325	YES	YES	YES
TI'KO PS325C	YES	YES	YES
TI'KO PS340C	YES	YES	YES
TI'KO PS420C	YES	YES	YES
TI'KO PS433C	YES	YES	YES
TI'KO PS450C	YES	YES	YES
TOSHIBA T1800	YES	N/A	N/A
TOSHIBA T1850	YES	N/A	N/A
TOSHIBA T1850C	YES	N/A	N/A
TOSHIBA T2000SX	YES	YES	YES
TOSHIBA T2000SXe	YES	YES	YES
TOSHIBA T2200SX	YES	YES	YES

Table 41 (Page 13 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)			
PCM MODEL	**OS/2 2.x**	**ES 1.0**	**LS 2.0**
TOSHIBA T3100SX	YES	YES	YES
TOSHIBA T3200SX	YES	YES	YES
TOSHIBA T3200SXC	YES	YES	YES
TOSHIBA T3300SL	YES	YES	YES
TOSHIBA T4400SX	YES	YES	YES
TOSHIBA T4400SXC	YES	YES	YES
TOSHIBA T5200	YES	YES	YES
TOSHIBA T5200/100	YES	YES	YES
TOSHIBA T6400SX	YES	YES	YES
TOSHIBA T6400SXC	YES	YES	YES
TOSHIBA T6400DX	YES	YES	YES
TOSHIBA T6400DXC	YES	YES	YES
TOSHIBA T8500 Model 25	YES	YES	YES
TOUCH TD333	YES	YES	YES
TOUCH TD433	YES	YES	YES
TRIGEM COMPUTERS TG SX 486 MM	YES	YES	YES
TULIP dc486 sx	YES	YES	YES
TULIP de486 dx/e	YES	YES	YES
TULIP de486 dx/50	YES	YES	YES
TULIP de486 dx/66	YES	YES	YES
TULIP Vision Line dt-486sx	YES	YES	YES
TULIP Vision Line dt-486dx/e	YES	YES	YES
TULIP Vision Line de-486sx/e	YES	YES	YES
TWINHEAD Slimnote 425sx	YES	YES	YES
TWINHEAD SS 600/425 C	YES	YES	YES
TWINHEAD Superpro 900E	YES	YES	YES
ULTRA-COMP 486DX - 33 I	YES	YES	YES
ULTRA-COMP 486DX - 50 I	YES	YES	YES
ULTRA-COMP 486DX2 - 66 I	YES	YES	YES
UNISYS MPE 46681	YES	YES	YES
UNISYS MPE 46681 WB	YES	YES	YES
UNISYS MPE 46681 WT	YES	YES	YES
UNISYS PC1-32532	YES	YES	YES
UNISYS PW2-4163-SX	YES	YES	YES
UNISYS PW2-Advantage 4336DX	YES	YES	YES
VICTOR V386M/33	YES	YES	YES
VICTOR V386MX/20	YES	YES	YES
VICTOR V486M/33	YES	YES	YES
VICTOR V486MX/20	YES	YES	YES
VIGLEN Genie Executive 4DX33	YES	YES	YES
VIGLEN Genie Micro 4SX20	YES	YES	YES
WANG Microsystems DTE33	YES	YES	YES
WANG Microsystems DTI250	YES	YES	YES
WANG Microsystems DTI66	YES	YES	YES
WANG Microsystems PC-350-40C	YES	YES	YES
WANG Microsystems EC 480/33C	YES	YES	YES
WEARNES Boldline 385SX-25	YES	YES	YES
WYSE TECHNOLOGY 6000i 645	YES	YES	YES

Table 41 (Page 14 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
WYSE TECHNOLOGY 7000i 740MP-INT	YES	YES	YES
WYSE TECHNOLOGY 7000i 760MP-INT	YES	YES	YES
WYSE TECHNOLOGY Decision 486SE-25SX	YES	YES	YES
WYSE TECHNOLOGY Decision 486SI-25SX	YES	YES	YES
WYSE TECHNOLOGY Decision 486/33E	YES	YES	YES
WYSE TECHNOLOGY Decision 486SE-33SX	YES	YES	YES
WYSE TECHNOLOGY Decision 486SI-33SX	YES	YES	YES
WYSE TECHNOLOGY Decision 486SE-33DX	YES	YES	YES
WYSE TECHNOLOGY Decision 486SI-33DX	YES	YES	YES
WYSE TECHNOLOGY Decision 486SE-50DX2	YES	YES	YES
WYSE TECHNOLOGY Decision 486SI-50DX2	YES	YES	YES
WYSE TECHNOLOGY Decision 486SI-66DX2	YES	YES	YES
Xtend Renaissance System Upgrade 386SX/20 PS/2 50	YES	YES	YES
Xtend Renaissance System Upgrade 386SX/20 PS/2 50Z	YES	YES	YES
Xtend Renaissance System Upgrade 386SX/20 PS/2 60	YES	YES	YES
Zenith Data Systems MastersPORT 386SL	YES	YES	YES
Zenith Data Systems MastersPORT 386SLe	YES	YES	YES
Zenith Data Systems Z-325/SX	YES	YES	YES
Zenith Data Systems Z-386/25	YES	YES	YES
Zenith Data Systems Z-386/33E	YES	YES	YES
Zenith Data Systems Z-425SX	YES	YES	YES
Zenith Data Systems Z-486SX/20E	YES	YES	YES
Zenith Data Systems Z-486SX/25E	YES	YES	YES
Zenith Data Systems Z-486/25E	YES	YES	YES
Zenith Data Systems Z-486/33E	YES	YES	YES
Zenith Data Systems Z-486/33ET	YES	YES	YES
Zenith Data Systems Z-NOTE 325L	YES	YES	YES
Zenith Data Systems Z-NOTE 325Lp	YES	YES	YES
Zenith Data Systems Z-NOTE 425Ln	YES	YES	YES
Zenith Data Systems Z-SERVER	YES	YES	YES
Zenith Data Systems Z-STATION 420Sn	YES	YES	YES
Zenith Data Systems Z-STATION 425Sh	YES	YES	YES
Zenith Data Systems Z-STATION 433DEh	YES	YES	YES
Zenith Data Systems Z-STATION 450XEn	YES	YES	YES
ZEOS 386SX-208DT	YES	YES	YES
ZEOS 386DX-33CT	YES	YES	YES
ZEOS 486-33/8	YES	YES	YES
ZEOS 486SLC	YES	YES	YES
ZEOS 486DX-33T	YES	YES	YES
ZEOS 486DX-33U	YES	YES	YES
ZEOS 486DX-50C	YES	YES	YES
ZEOS 486DX2-50U	YES	YES	YES
ZEOS 486DX2-66CU	YES	YES	YES
ZEOS 486DX2-66E	YES	YES	YES
ZEOS 486DX2-66U	YES	YES	YES
ZEOS Upgradeable 486-33C	YES	YES	YES
ZEOS Upgradeable 486DX2-66	YES	YES	YES
ZEOS Upgradeable Local Bus - i486DX2-66	YES	YES	YES

Table 41 (Page 15 of 15). PCM Systems Compatibility Matrix (as of April 27th 1993)

PCM MODEL	OS/2 2.x	ES 1.0	LS 2.0
ZEOS Upgradeable Local Bus - IDE	YES	YES	YES
ZEOS Upgradeable Local Bus - SCSI	YES	YES	YES
TOTAL PCMs 110 - MODELS PASSING TESTS	***674***	***664***	***664***

Table notes:

N/A No Adapter Slots Available

Appendix C. OS/2 2.x Compatible Hardware Devices

OS/2 2.1 has been tested in conjunction with many PC adapters and peripherals from a wide range of OEM manufacturers and Independent Hardware Vendors (IHVs), by IBM at its OEM test laboratories in Boca Raton, Florida and Basingstoke, England, and also via two widespread beta tests of OS/2 2.1.

The current list of OEM and IHV hardware supported by OS/2 2.1 is included in the **PCMTABLE** package, available on CompuServe, and to IBM internal users on OS2TOOLS.

These lists are maintained by IBM's Worldwide Industry Hardware Support group in the Personal Systems Programming laboratory at Boca Raton in Florida.

The contents of these hardware support lists, as of May 10th 1993, has been reproduced in this appendix.

If a PC hardware adapter or peripheral is not in this appendix, please check CompuServe for a more recent version of these lists, or contact the hardware manufacturer.

C.1 Introduction to Compatibility Tables for Hardware Devices

The following tables specify hardware devices tested and found to be compatible with IBM OS/2 2.0, OS/2 2.0 with Service Pak XR06055 applied, or OS/2 2.1 The tables indicate at which release these devices were tested as compatible or what vendor provides support for these devices. The release numbers indicate support as of that release with support continuing into subsequent releases unless otherwise indicated.

Not all of the products listed have been tested by IBM. Many have been tested by the respective hardware vendors or a third party.

Note

Many of the products listed require device drivers that are not included with OS/2. Availbility and support of such drivers is normally through the hardware vendor or a third party.

Presence on this list does not necessarily mean that the product listed is compatible with OS/2 in your system configuration. When in doubt, you should seek additional information from your dealer or the appropriate vendor.

The compatibility test information provided herein is provided for information purposes. IBM makes no warranty, express or implied, with respect to the operation of OS/2 on the personal computers, devices, or adapters listed.

C.2 Display Adapters

Table 42. Display Adapters (as of May 10th 1993)

Vendor	Product or Chip Name	Available From/As of
ATI	Graphics Ultra 8514/A (Uses IBM 8514 Driver)	OS/2 2.0
ATI	ATI 28800 (chip)	OS/2 2.1
ATI	(See Vendor for Additional Product Names)	ATI
Boca Research, Inc.	SuperVga	OS/2 2.0 Service Pak
Cirrus Logic, Inc.	CL-GD5422/24 (chip)	OS/2 2.1
Cirrus Logic, Inc.	(See Vendor for Names)	Cirrus Logic, Inc.
COMPAQ	QVision	OS/2 2.0 Service Pak
COMPAQ	Prolinea 486 Family (Tseng ET4000)	OS/2 2.0 Service Pak
Diamond Computer Systems, Inc.	SpeedStar SuperVGA	OS/2 2.0 Service Pak
Everex System, Inc.	Viewport NI	OS/2 2.0 Service Pak
Headland Technology, Inc.	VideoSeven (HT209 chip)	OS/2 2.1
IBM	8514 16 Bit/32 Bit Seamless	OS/2 2.0/OS/2 2.1
IBM	Action Media II Display/Monitor Adapter w/2MB	OS/2 2.0
IBM	ActionMedia II Display/Monitor Adapter/A w/2MB	OS/2 2.0
IBM	Display Monitor Adapter/A XGA-2 w/1MB	OS/2 2.0 Service Pak
IBM	Extended Graphics Adapter/A (XGA)	OS/2 2.0
IBM	IBM VGA256C (Mod 700C, PS/2 25SX, PS/1-2135)	OS/2 2.0 Service Pak
Orchid Technology, Inc.	ProDesigner II/MC	OS/2 2.0 Service Pak
Sigma Designs, Inc.	Sigma VGA Legend II	OS/2 2.0 Service Pak
STB Systems, Inc.	Ergo-VGA/MC	OS/2 2.0 Service Pak
STB Systems, Inc.	PowerGraph VGA	OS/2 2.0 Service Pak
Trident Microsystems, Inc	Trident TVGA8900B/C (chip)	OS/2 2.1
Trident	(See Vendor for Additional Product Names)	Trident
Tseng Labs	ET4000 (chip)	OS/2 2.0 Service Pak
VGA Graphic Card	JAX-8212	OS/2 2.1
Western Digital Corp.	Paradise	OS/2 2.1
Western Digital Corp.	WD90C11 (chip)	OS/2 2.1
Western Digital Corp.	WD90C30 (chip)	OS/2 2.1
Western Digital Corp.	WD90C31 (chip) (in WD90C30 Compability Mode)	OS/2 2.1
Wyse Technology, Inc.	Amdek SmartVision/SVGA	OS/2 2.0 Service Pak

Note:

1. OS/2 2.0 supports EGA, CGA and VGA adapters. (Any VGA adapter 100% compatabile with the IBM VGA adapter is supported.)
2. OS/2 supplies generic drivers for the documented chip sets. Where only chips are listed, specific boards/OEM systems have been tested with these chips. However, some vendors change external clock chips and graphics chips without renaming the boards or system models. When in doubt, you should seek additional information from the appropriate vendor.

C.3 SCSI Adapters

Table 43 (Page 1 of 3). SCSI Adapters (as of May 10th 1993)

Vendor	Product Name	Available From/As of
Adaptec	AIC 6260	OS/2 2.0
Adaptec	AHA 1510	OS/2 2.0
Adaptec	AHA 1520/1522	OS/2 2.0
Adaptec	AHA 1540/1542A,B,C	OS/2 2.0

Table 43 (Page 2 of 3). SCSI Adapters (as of May 10th 1993)

Vendor	Product Name	Available From/As of
Adaptec	AHA 1640	OS/2 2.0
Adaptec	AHA 1740/1742/1742A/1744 (Standard Mode)	OS/2 2.0
Adaptec	AHA 1740/1742/1742A/1744 (Enhanced Mode)	OS/2 2.0
Adaptec	AIC 7770	Adaptec
Always Technology	Always Technology IN-2000 SCSI Adapter	Always
AT&T	SCSI Host Bus Adapter	OS/2 2.0
BusLogic, Inc.	BT-542B & D ISA SCSI Adapter	Bus Logic
BusLogic, Inc.	BT-545S ISA with FAST SCSI Adapter	BusLogic
BusLogic, Inc.	BT-640A MCA SCSI Adapter	BusLogic
BusLogic, Inc.	BT-646S & D MCA FAST SCSI Adapter	BusLogic
BusLogic, Inc.	BT-742A EISA SCSI Adapter	BusLogic
BusLogic Inc.	BT747S & D EISA with FAST SCSI Adapter	BusLogic
BusLogic, Inc.	BT-757S & D EISA with FAST SCSI Adapter	BusLogic
BusLogic, Inc.	BT-445S SCSI Adapter	BusLogic
Ciprico, Inc.	5500 ISA Host Bus Adapter	Ciprico, Inc.
Ciprico, Inc.	5600 EISA Host Bus Adapter	Ciprico, Inc.
Ciprico, Inc.	5700 Micro Channel Host Bus Adapter	Ciprico, Inc.
Compaq	Intelligent Drive Array (IDA)	Compaq
Compaq	Intelligent Drive Array-2 (IDA-2)	Compaq
Compaq	Intelligent Array Expansion System (IAES)	Compaq
Compaq	32 Bit Fast SCSI-2 Controller	Compaq
Computer Elektronik Infosystems	C5630 AT-Bus Adapter	Computer Electronik Infosystems
Computer Elektronik Infosystems	C5640 Micro Channel Adapter	Computer Electronik Infosystems
DPT	PM2011 SmartCache Plus ISA SCSI Controller	DPT, OS/2 2.1
DPT	PM2012 SmartCache Plus EISA SCSI Controller	DPT, OS/2 2.1
Future Domain	TMC-845 (No Longer Shipping, Replaced by TMC-850MEX)	OS/2 2.0
Future Domain	TMC-875 (No Longer Shipping, Replaced by TMC-885M)	OS/2 2.0
Future Domain	TMC-850M/850MER/850MEX	OS/2 2.0
Future Domain	TMC-860M/885M	OS/2 2.0
Future Domain	TMC-850IBM	OS/2 2.0
Future Domain	TMC-1650/1670	OS/2 2.0
Future Domain	TMC-1660/1680	OS/2 2.0
Future Domain	TMC-1790/1795	OS/2 2.1
Future Domain	MCS-600/700	OS/2 2.0
Future Domain	TMC-7000EX (EISA) Adapter	OS/2 2.0
Future Domain	TMC-950 (Chip)	OS/2 2.0
Future Domain	TMC-1800/18C50 (Chip)	OS/2 2.0
IBM	IBM PS/2 Micro Channel SCSI Adapter-1018	OS/2 2.0
IBM	IBM PS/2 Mirco Channel SCSI Adapter/A with Cache-1005	OS/2 2.0
IBM	IBM 16 Bit AT FAST SCSI Adapter	OS/2 2.1
Mylex	DCE376 MCA Disk Array Controller	Mylex BBS
Mylex	DAC960 EISA Disk Array Controller	Mylex BBS
Procomm	MCA Enabler Adapter	Procomm
Procomm	ISA SCSI Accelerator	Procomm
Trantor	TSOS2 ADD Adapter	Trantor, CompuServ, Genie, Online America
Trantor	T128/130/338/348	Trantor

Table 43 (Page 3 of 3). SCSI Adapters (as of May 10th 1993)		
Vendor	**Product Name**	**Available From/As of**
Ultrastor	ULTRA 14F ISA Busmaster SCSI-2	Ultrastor
Ultrastor	ULTRA 24F EISA Busmaster SCSI-2	Ultrastor
Ultrastor	ULTRA 34F VL-Bus Busmaster SCSI-2	Ultrastor
Ultrastor	ULTRA 124F EISA Busmaster Array	Ultrastor

Note:

- Many disk devices are supported partially or fully by the generic OS/2 2.0 disk device driver (ie., any Controller 100% Compatible with Western Digital 1003 (IDE, ESDI)).

C.4 CD-ROM Drives (SCSI-based)

Table 44. CDROM Drives (SCSI-based) (as of May 10th 1993)		
Vendor	**Product Name**	**Available From/As of**
Compaq	Dual Speed CDROM	Compaq
Hitachi	CDR-1650S	OS/2 2.1
Hitachi	CDR-1750S	OS/2 2.1
Hitachi	CDR-3650	OS/2 2.1
Hitachi	CDR-3750	OS/2 2.1
IBM	IBM CD-ROM I	OS/2 2.0
IBM	IBM CD-ROM II	OS/2 2.0
NEC	CDR-25	OS/2 2.1
NEC	CDR-36	OS/2 2.1
NEC	CDR-37	OS/2 2.1
NEC	CDR-38	OS/2 2.1
NEC	CDR-72	OS/2 2.1
NEC	CDR-73	OS/2 2.1
NEC	CDR-74	OS/2 2.1
NEC	CDR-82	OS/2 2.1
NEC	CDR-83	OS/2 2.1
NEC	CDR-84	OS/2 2.1
Panasonic	CR-501	OS/2 2.1
Panasonic	LK-MC501S	OS/2 2.1
Pioneer	DRM-600 (Data Access Only)	OS/2 2.1
Pioneer	DRM-604X	OS/2 2.1
Sony	CDU-541	OS/2 2.1
Sony	CDU-561	OS/2 2.1
Sony	CDU-6111	OS/2 2.1
Sony	CDU-6211	OS/2 2.1
Sony	CDU-7211	OS/2 2.1
Texel	DM-3021	OS/2 2.1
Texel	DM-3024	OS/2 2.1
Texel	DM-5021	OS/2 2.1
Texel	DM-5024	OS/2 2.1
Toshiba	Toshiba XM-3101	OS/2 2.0
Toshiba	Toshiba XM-3201	OS/2 2.0
Toshiba	Toshiba XM-3301	OS/2 2.0
Toshiba	Toshiba XM-3401	OS/2 2.1

Table 45. SCSI Adaptors Tested for Use with above SCSI CDROM Drives (as of May 10th 1993)

Vendor	Product Name	Available From/As of
Adaptec	AIC 6260	OS/2 2.1
Adaptec	AHA 1510/1520/1522	OS/2 2.1
Adaptec	AHA 1540/1542	OS/2 2.1
Adaptec	AHA 1640	OS/2 2.1
Adaptec	AHA 1740/1742/1744 Standard and Enhanced Modes	OS/2 2.1
DPT	PM2011/2012	OS/2 2.1
Future Domain	TMC-845/850/860/875/885	OS/2 2.1
Future Domain	TMC-1650/1660/1670/1680	OS/2 2.1
Future Domain	MCS-600/700	OS/2 2.1
Future Domain	TMC-7000EX	OS/2 2.1
Future Domain	TMC-850IBM	OS/2 2.1
IBM	PS/2 SCSI Adapter	OS/2 2.1
IBM	PS/2 SCSI Adapter with Cache	OS/2 2.1

Note:

This table specifies SCSI hardware devices that IBM has tested and found to be compatible with the above CDROM Drives. Other adaptors will work, but restrictions may exist. If in doubt please contact the manufacturer of the adaptor or CDROM drive. Information on restrictions is printed in the readme materials shipped with the operating system and represents the best known information at that time.

C.5 Miscellaneous Storage Support

Table 46. OS/2 Miscellaneous Storage Support (as of May 10th 1993)

Company	Product Name	Support From/As of
Iomega	Bernoulli Transportable 44	OS/2 2.0
Iomega	Bernoulli Transportable 90	OS/2 2.0
Quantum	Passport XL	OS/2 2.0
Quantum	Hardcard EZ	OS/2 2.0
Quantum	Hardcard II XL 105	OS/2 2.0
Syquest	SQ 800	OS/2 2.0
Integra Technologies	Oasas I RAID Disk Array	Oasas I Ver 2
IBM	IBM 3995 (Jukebox)	Driver Shipped with Hardware
IBM	IBM MD3125 (3.5") (Read/Write Optical Drive)	Driver Shipped with Hardware
IBM	IBM MD3431 (5 1/2') (Read/Write Optical Drive)	Driver Shipped with Hardware

C.6 Keyboards

Table 47. Keyboards (as of May 10th 1993)

Vendor	Product Name	Available As of
IBM	IBM Enhanced 101-Key Keyboard	OS/2 2.0

Note:

Any keyboard compatible with the IBM Enhanced 101-Key Keyboard will be supported.

C.7 Pointing Devices

Table 48. Pointing Devices (as of May 10th 1993)

Vendor	Product Name	Available As of
IBM	IBM PS/2 Mouse	OS/2 2.0
IBM	IBM 8516 Touch Display	OS/2 2.0
Kensington Microware, Ltd	Kensington Expert PS/2 Mouse	OS/2 2.0
Logitech, Inc.	Logitech PS/2 Mouse	OS/2 2.0
Logitech, Inc.	Logitech Serial Mouse (Series C)	OS/2 2.0
Logitech, Inc.	Logitech Serial Mouse (Series M)	OS/2 2.0
Logitech, Inc.	Logitech Trackman Serial Mouse	OS/2 2.0
Microsoft	MS Bus Mouse	OS/2 2.0
Microsoft	MS Inport Mouse	OS/2 2.0
Microsoft	MS PS/2 Mouse	OS/2 2.0
Mouse Systems	PC Mouse Systems Serial Mouse	OS/2 2.0
Mouse Systems	PC Mouse Systems Bus Mouse	OS/2 2.0

Note:

1. Any pointing device that is compatible with the above devices will be supported.
2. As of OS/2 2.1, enhanced support is provided for bus/serial mice. The mouse driver will now auto-detect mice on any serial port in the system. Once found, it will auto-detect the interrupt level the mouse is on. Previously, only serial ports that were known to the machine's BIOS were supported.
3. Previous support was only for IRQ 3 & 4. OS/2 2.1 now scans IRQ's 3,4,5,7,12, and 15.

C.8 Scanners

Table 49 (Page 1 of 2). Scanners (as of May 10th 1993)

Vendor	Product Name	Available From
Bell & Howell	Copiscan II Model 2135	IBM SAA ImagePlus/2 IBM SAA ImagePlus Workstation Program/2
Bell & Howell	Copiscan II Model 3338	IBM SAA ImagePlus/2 IBM SAA ImagePlus Workstation Program/2
Eastman Kodak	IMAGELINK Scanner 900	Kodak for use with IBM SAA ImagePlus Workstation Program/2
Hewlett-Packard	ScanJet	IBM OS/2 Image Support V1.1
Hewlett-Packard	ScanJet Plus	IBM OS/2 Image Support V1.1
Howtek	ScanMaster	IBM OS/2 Image Support V1.1 IBM SAA ImagePlus Workstation Program/2
Howtek	ScanMaster II	IBM OS/2 Image Support V1.1 IBM SAA ImagePlus Workstation Program/2
IBM	3117 Scanner	IBM OS/2 Image Support V1.1
IBM	3118 Scanner	IBM OS/2 Image Support V1.1
IBM	3119 PageScanner	IBM OS/2 Image Support V1.1 IBM SAA ImagePlus/2 IBM SAA ImagePlus Workstation Program/2
IBM	2456 Scanner	IBM SAA ImagePlus/2 IBM SAA ImagePlus Workstation Program/2

Table 49 (Page 2 of 2). Scanners (as of May 10th 1993)		
Vendor	**Product Name**	**Available From**
Polaroid	CS-500	IBM SAA ImagePlus Workstation Program/2
Sharp Electronics	JX-300	IBM OS/2 Image Support V1.1
Sharp Electronics	JX-450	IBM OS/2 Image Support V1.1 IBM SAA ImagePlus Workstation Program/2
Terminal Data Corporation	DS-2610W	TDC for use with: IBM SAA ImagePlus Workstation Program/2
Note: This table lists OS/2 mode support. Most scanners/drivers designed to operate under DOS or Windows will operate in OS/2 DOS or WIN-OS/2 modes.		

C.9 Multimedia Adapters

Table 50. OS/2 Multimedia Adapter Support (as of May 10th 1993)		
Vendor	**Product Name**	**Available From/As of**
Creative Labs	Soundblaster	OS/2 2.1, MMPM/2 1.1
Creative Labs	Soundblaster Pro (ISA)	OS/2 2.1, MMPM/2 1.1
Creative Labs	Soundblaster Pro MCA	OS/2 2.1, MMPM/2 1.1
Creative Labs	Soundblaster 16 ASP	OS/2 2.1, MMPM/2 1.1
IBM	IBM Action Media/2 Playback Adapter (ISA, MCA)	Driver Shipped with Card
IBM	IBM Action Media/2 Capture Adapter (Pre-req Action Media/2 Playback Adapter)	Driver Shipped with Card
IBM	M-Audio Capture and Playback Adaptor (ISA, MCA)	OS/2 2.1, MMPM/2 1.1
IBM	IBM M-Motion Video Adapter/A (MCA)	M-Control Program/2 Ver 2.01
MediaVision	Proaudio Spectrum 16	OS/2 2.1, MMPM/2 1.1

C.10 Printers and Plotters

Table 51 (Page 1 of 9). Printers and Plotters (as of May 10th 1993)		
Vendor	**Device Description**	**Available From/As of**
AST	AST TurboLaser	OS/2 2.0 - (PSCRIPT.DRV)
Agfa	Agfa Matrix ChromaScript v51_8	OS/2 2.0 - (PSCRIPT.DRV)
Agfa	Agfa-Compugraphic 9400PS v49_3	OS/2 2.0 - (PSCRIPT.DRV)
Agfa	Agfa/Compugraphic 400PS	OS/2 2.0 - (PSCRIPT.DRV)
Apple	Apple LaserWriter II NT	OS/2 2.0 - (PSCRIPT.DRV)
Apple	Apple LaserWriter II NTX	OS/2 2.0 - (PSCRIPT.DRV)
Apple	Apple LaserWriter Plus v42_2	OS/2 2.0 - (PSCRIPT.DRV)
Apple	Apple LaserWriter Plus	OS/2 2.0 - (PSCRIPT.DRV)
Apple	Apple LaserWriter	OS/2 2.0 - (PSCRIPT.DRV)
Canon	Canon BubbleJet BJ10E (Proprinter X24E)	OS/2 2.0 - (IBM42XX.DRV) also Canon BBS (714) 438-3325
Canon	Canon BubbleJet BJC800 (Use Epson LQ-2550)	OS/2 2.0 - (EPSON.DRV) also Canon BBS (714) 438-3325
Canon	Canon BubbleJet LBP8 III+ (Use Epson LQ-2550)	OS/2 2.0 - (EPSON.DRV) also Canon BBS (714) 438-3325

Table 51 (Page 2 of 9). Printers and Plotters (as of May 10th 1993)

Vendor	Device Description	Available From/As of
Citizen	Citizen PN48	OS/2 2.1 - (EPSON.DRV)
Compaq	Compaq Pagemarq 20	OS/2 2.1 - (PSCRIPT.DRV)
Dataproducts	Dataproducts LZR 1260 v47_0	OS/2 2.0 - (PSCRIPT.DRV)
Dataproducts	Dataproducts LZR-2665	OS/2 2.0 - (PSCRIPT.DRV)
Digital	Digital LN03R ScriptPrinter	OS/2 2.0 - (PSCRIPT.DRV)
Digital	Digital LPS PrintServer 40	OS/2 2.0 - (PSCRIPT.DRV)
Epson (Generic)	Epson 24 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson (Generic)	Epson 24 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson (Generic)	Epson 9 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson (Generic)	Epson 9 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson AP-800 - Color 48 elements - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson AP-2000 - 9 pin (Use Epson LX810)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson AP-2250 - 9 pin 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson AP-2500 - 9 pin (Use Epson FX-1050)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson AP-3000 - 24 pin (Use Epson LQ-200)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson AP-3250 - 24 pins 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson AP-4000 - 24 pins (Use Epson LQ-510)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson AP-4500 - 24 pins (Use Epson LQ-1010)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson AP-5000	OS/2 2.1 - (EPSON.DRV)
Epson	Epson AP-5500 - 24 pins	OS/2 2.1 - (EPSON.DRV)
Epson	Epson Actionlaser 1000	OS/2 2.1 - (LASERJET.DRV)
Epson	Epson Actionlaser 1500	OS/2 2.1 - (LASERJET.DRV)
Epson	Epson Actionlaser II	OS/2 2.1 - (LASERJET.DRV)
Epson	Epson Actionlaser(+) - (EPL-6000, Laserjet II)	OS/2 2.0 - (LASERJET.DRV)
Epson	Epson Apex 80 - 9 pin - (Use Epson LX-800)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson Citizen PN48	OS/2 2.1 - (EPSON.DRV)
Epson	Epson DFX-5000 - 9 pin 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson DFX-8000 9 pin - 136 column	OS/2 2.0 - (EPSON.DRV)
Epson	Epson DLQ-2000 24 pins - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson EPL-6000 Laser	OS/2 2.0 - (EPSON.DRV)
Epson	Epson EPL-7000	OS/2 2.0 - (LASERJET.DRV)
Epson	Epson EPL-7500 v52_3	OS/2 2.0 - (PSCRIPT.DRV)
Epson	Epson EPL-8000	OS/2 2.1 - (LASERJET.DRV)
Epson	Epson EPL-8000 PS Card 82605	OS/2 2.1 - (PSCRIPT.DRV)
Epson	Epson EX-1000 Color 9 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson EX-800 Color 9 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson FX-1000 9 pins - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson FX-1050 9 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson FX-1170	OS/2 2.1 - (EPSON.DRV)
Epson	Epson FX-286e 9 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson FX-850 9 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson GQ-3500 Laser	OS/2 2.1 - (EPSON.DRV)
Epson	Epson GQ-5000 Laser	OS/2 2.11111EPSON.DRV)
Epson	Epson JX-80 Color 9 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson L-750 24 pins - 80 columns (Use Epson LQ-500)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson L-1000 24 pins - 80 columns (Use Epson LQ-500)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-400 24 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-500 24 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)

Table 51 (Page 3 of 9). Printers and Plotters (as of May 10th 1993)		
Vendor	**Device Description**	**Available From/As of**
Epson	Epson LQ-510 24 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-550 (N9) 24 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-570 24 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-850 24 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-850 plus 24 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-860 Color 24 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-870 24 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-950 (N9) 24 pins - 110 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-1010 24 pin - 132 column	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-1050 (N9) 24 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-1050 24 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-1050 plus 24 pins - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-1060 Color 24 pins - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-1070	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LQ-1170 24 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-2500 Color 24 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LQ-2550 Color 24 pins - 136 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LX-400 9 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson LX-800 9 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LX-810 9 pins - 80 columns	OS/2 2.0 - (EPSON.DRV)
Epson	Epson LX-850 9 pins - 80 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson SQ-850 24 pins - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson SQ-2500 24 Nozzle Inkjet - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson SQ-2550 24 Nozzle Inkjet - 136 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson Stylus 800	OS/2 2.1 - (EPSON.DRV)
Epson	Epson T-750 9 pins (Use Epson FX-286e)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson T-1000 9 pins (Use Epson LX-800)	OS/2 2.0 - (EPSON.DRV)
Epson	Epson TLQ-4800 48 pins 143 columns	OS/2 2.1 - (EPSON.DRV)
Epson	Epson TSQ-4800 48 Nozzle Inkjet - 143 columns	OS/2 2.1 - (EPSON.DRV)
Hewlett Packard	HP 7440A Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7470A Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7475A Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7550A Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7580A Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7580B Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7585A Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7585B Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP 7586B Plotter	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP DeskJet	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DeskJet Plus	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DeskJet Portable	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DeskJet 500	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DeskJet 500C	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DeskJet 510C	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DeskJet 550C	OS/2 2.1 - (HPDJPM.DRV)
Hewlett Packard	HP DraftMaster I - (HP-7595A)	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP DraftMaster II - (HP-7596A)	OS/2 2.0 - (PLOTTERS.DRV)
Hewlett Packard	HP DraftPro (HP-7570A)	OS/2 2.0 - (PLOTTERS.DRV)

Table 51 (Page 4 of 9). Printers and Plotters (as of May 10th 1993)

Vendor	Device Description	Available From/As of
Hewlett Packard	HP LaserJet 4	OS/2 2.1 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet 4L	OS/2 2.1 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet 4M	OS/2 2.1 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet 4/4M PS v2011_110	OS/2 2.1 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet Classic	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet 500 Plus	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet 2000	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet IID v52_2	OS/2 2.0 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet IID	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet IIP	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet IIP Plus	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet IIP v52_2	OS/2 2.0 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet III	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet III Cartridge Plus	OS/2 2.1 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet III v52_2	OS/2 2.0 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet IIID	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet IIID v52_2	OS/2 2.0 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet IIID Cartridge Plus	OS/2 2.1 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet IIIP PS v52_2	OS/2 2.0 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet IIIP	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet IIISi PS v52_3	OS/2 2.0 - (PSCRIPT.DRV)
Hewlett Packard	HP LaserJet IIISi	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet Plus	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP LaserJet Series II	OS/2 2.0 - (LASERJET.DRV)
Hewlett Packard	HP PaintJet	OS/2 2.0 - (SMGXPJET.DRV)
Hewlett Packard	HP PaintJet XL	OS/2 2.0 - (SMGXPJET.DRV)
Hewlett Packard	HP PaintJet XL300 PS v2011_112	OS/2 2.1 (PSCRIPT.DRV)
IBM	IBM 2380 PPS II	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 2381 PPS II	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 2390 PPS II	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 2391 PPS II	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 3812 Page Printer (Use IBM 5152 Graphics Printer)	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM5152.DRV)
IBM	IBM 3816 - 01D	OS/2 2.0 - (IBM52XX.DRV)
IBM	IBM 3816 - 01S	OS/2 2.0 - (IBM52XX.DRV)
IBM	IBM 3852 Color Inkjet	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM3852.DRV)
IBM	IBM 4019 LaserPrinter	OS/2 2.0 - (IBM4019.DRV)
IBM	IBM 4019 Laserprinter	OS/2 2.0 - (LASERJET.DRV)
IBM	IBM 4019 LaserPrinter E	OS/2 2.0 - (IBM4019.DRV)
IBM	IBM 4019 Laserprinter E	OS/2 2.0 - (LASERJET.DRV)
IBM	IBM 4019 v52_1 (17 Fonts)	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4019 v52_1 (39 Fonts)	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4029 (17 Fonts 300 Dpi)	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4029 (17 Fonts 600 Dpi)	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4029 (39 Fonts 300 Dpi)	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4029 (39 Fonts 600 Dpi)	OS/2 2.0 - (PSCRIPT.DRV)

Table 51 (Page 5 of 9). Printers and Plotters (as of May 10th 1993)

Vendor	Device Description	Available From/As of
IBM	IBM 4029 LaserPrinter 5E	OS/2 2.0 - (IBM4019.DRV) or (LASERJET.DRV)
IBM	IBM 4029 LaserPrinter 6	OS/2 2.0 - (IBM4019.DRV) or (LASERJET.DRV)
IBM	IBM 4029 LaserPrinter 6P	OS/2 2.1 - (IBM4019.DRV)
IBM	IBM 4029 LaserPrinter 10	OS/2 2.0 - (IBM4019.DRV) or (LASERJET.DRV)
IBM	IBM 4029 LaserPrinter 10L	OS/2 2.0 - (IBM4019.DRV) or (LASERJET.DRV)
IBM	IBM 4029 LaserPrinter 10P	OS/2 2.1 - (IBM4019.DRV)
IBM	IBM 4039 LaserPrinter 300 DPI	OS/2 2.1 - (PSCRIPT.DRV)
IBM	IBM 4039 LaserPrinter 600 DPI	OS/2 2.1 - (PSCRIPT.DRV)
IBM	IBM 4070 IJ	OS/2 2.1 - (IBM42XX.DRV)
IBM	IBM 4072 ExecJet Printer	OS/2 2.1 - (IBM42XX.DRV)
IBM	IBM 4079 Color Jetprinter PS 4079	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4201 Proprinter II	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4201 Proprinter III	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4201 Proprinter	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4202 Proprinter II XL	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4202 Proprinter III XL	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4202 Proprinter XL	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4207 Proprinter X24	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4207 Proprinter X24E	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4208 Proprinter XL24	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4208 Proprinter XL24E	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4216-031 v51_4 SheetFeed	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 4224 - 01 & 02 & E3	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4224 - C2	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 4226 Model 302	OS/2 2.0 - (IBM42XX.DRV)
IBM	IBM 5183 Portable Printer	OS/2 2.1 - (EPSON.DRV)
IBM	IBM 5201 Quietwriter	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM52011.DRV)
IBM	IBM 5201 Quietwriter II	OS/2 2.0 - (IBM52012.DRV)
IBM	IBM 5202 QuietWriter III	OS/2 2.0 - (IBM52XX.DRV)
IBM	IBM 5204 QuickWriter	OS/2 2.0 - (IBM52XX.DRV)
IBM	IBM 5152 Graphics Printer	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM5152.DRV)
IBM	IBM 5182 Color Printer	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM5182.DRV)
IBM	IBM 5216 Wheelwriter I	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM5216.DRV)
IBM	IBM 5216 Wheelwriter II	CompuServe, IBM P.C.C. BBS (404) 835-6600 (IBM5216.DRV)
IBM	IBM 6180 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 6182 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 6184 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 6186-1 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 6186-2 Plotter	OS/2 2.0 - (PSCRIPT.DRV)

Table 51 (Page 6 of 9). Printers and Plotters (as of May 10th 1993)

Vendor	Device Description	Available From/As of
IBM	IBM 7371 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 7372 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 7374 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 7375-1 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM 7375-2 Plotter	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM NULL Printer Driver	OS/2 2.0 - (IBMNULL.DRV)
IBM	IBM Personal Page Printer II-30	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM Personal Page Printer II-31	OS/2 2.0 - (PSCRIPT.DRV)
IBM	IBM Personal Pageprinter	OS/2 2.0 - (PSCRIPT.DRV)
Kyocera	Kyocera F-800A/F-800	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera F-820	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera F-1000A/F-1000	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera F-1800A/F-1800	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera F-2000A/F-2200S	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera F-3000A/F-3300	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera F-5000A/F-5000	OS/2 2.0 - (LASERJET.DRV)
Kyocera	Kyocera FS-850A/FS-850	OS/2 2.1 - (LASERJET.DRV)
Kyocera	Kyocera FS-1500A/FS-1500	OS/2 2.1 - (LASERJET.DRV)
Kyocera	Kyocera P-2000	OS/2 2.0 - (PSCRIPT.DRV)
Kyocera	Kyocera Q-8010	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 100 v38_0	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 100 v42_5	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 200 v47_1	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 200 v49_3	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 300 v47_0	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 300 v47_1	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 300 v49_3	OS/2 2.0 - (PSCRIPT.DRV)
Linotronic	Linotronic 500 v49_3	OS/2 2.0 - (PSCRIPT.DRV)
NEC Corp.	NEC Colormate PS v51_9	OS/2 2.0 - (PSCRIPT.DRV)
NEC Corp.	NEC LC-890	OS/2 2.0 - (PSCRIPT.DRV)
NEC Corp.	NEC P3200 Printer - (Use Epson LQ-850)	OS/2 2.0 - (EPSON.DRV)
NEC Corp.	NEC P6200 Printer - (Use Epson LQ-2550)	OS/2 2.0 - (EPSON.DRV)
NEC Corp.	Silentwriter LC 890XL v50_5	OS/2 2.0 - (PSCRIPT.DRV)
NEC Corp.	Silentwriter2 290 v52_0	OS/2 2.0 - (PSCRIPT.DRV)
NEC Corp.	Silentwriter2 Model 90 v52_2	OS/2 2.0 - (PSCRIPT.DRV)
Okidata	Okidata Microline 84 (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 84 Step II (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata PaceMark 2350 (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata PaceMark 2410 (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata PaceMark 3410 (Some models emulate IBM Proprinter or Epson FX)	OS/2 2.0 - (IBM42xx.DRV or (EPSON.DRV)
Okidata	Okidata Okimate 20 (Some models emulate IBM Proprinter)	OS/2 2.0 - (IBM42XX.DRV)
Okidata	Okidata 180 (Some models emulate Epson FX Printer)	OS/2 2.0 - (EPSON.DRV)
Okidata	Okidata Microline 172 (Some models emulate IBM 5152 Printer)	OS/2 2.0 - (IBM5152.DRV)

Table 51 (Page 7 of 9). Printers and Plotters (as of May 10th 1993)

Vendor	Device Description	Available From/As of
Okidata	Okidata Microline 182 (Some models emulate IBM 5152 Printer)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 182 Plus (Some models emulate IBM 5152 Printer)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 182 Turbo (Some models emulate IBM 5152 Printer)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 192 (Some models emulate IBM 5152 Printer)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 192 Plus (Some models emulate IBM 5152 or Proprinter)	OS/2 2.0 - (IBM5152.DRV or (IBM42XX.DRV)
Okidata	Okidata Microline 193 (Some models emulate IBM 5152 Printer)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 193 Plus (Some models emulate IBM 5152 or Proprinter)	OS/2 2.0 - (IBM5152.DRV or (IBM42XX.DRV)
Okidata	Okidata Microline 292 (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 293 (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 294 (Some models emulate IBM 5152)	OS/2 2.0 - (IBM5152.DRV)
Okidata	Okidata Microline 320 (Some models emulate IBM Proprinter or Epson-FX)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 321 (Some models emulate IBM Proprinter or Epson-FX)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 380 (Some models emulate Epson-LQ)	OS/2 2.0 - (EPSON.DRV)
Okidata	Okidata Microline 390 (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 390 Plus (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 391 (Some models emulate IBM Proprinter 24 pin or Epson-FX)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 391 Plus (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 393B (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 393B Plus (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 393C (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Microline 393C Plus (Some models emulate IBM Proprinter 24 pin or Epson-LQ)	OS/2 2.0 - (IBM42XX.DRV or (EPSON.DRV)
Okidata	Okidata Laserline 6 (Some models emulate HP LaserJet Plus)	OS/2 2.0 - (LASERJET.DRV)
Okidata	Okidata OL400 LED Pageprinter (Some models emulate HP LaserJet Plus)	OS/2 2.0 - (LASERJET.DRV)
Okidata	Okidata OL800 LED Pageprinter (Some models emulate HP LaserJet II or IBM proprinter)	OS/2 2.0 - (LASERJET.DRV or (IBM42XX.DRV)
Okidata	Okidata OL820 LED Pageprinter (Some models emulate HP LaserJet II or IBM proprinter)	OS/2 2.0 - (LASERJET.DRV or (IBM42XX.DRV)
Okidata	Okidata OL830 LED Pageprinter (Some models emulate HP LaserJet II or Generic Postscript)	OS/2 2.0 - (LASERJET.DRV or (PSCRIPT.DRV)
Okidata	Okidata OL840 LED Pageprinter (Some models emulate HP LaserJet II or Generic Postscript)	OS/2 2.0 - (LASERJET.DRV or (PSCRIPT.DRV)
Olivetti	Olivetti LP 5000	OS/2 2.0 - (PSCRIPT.DRV)
Panasonic	Panasonic KX-P1123 in Epson LQ-850 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P1124 in Epson LQ-2550 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P1124i in Epson LQ-850 mode	OS/2 2.0 - (EPSON.DRV)

Table 51 (Page 8 of 9). Printers and Plotters (as of May 10th 1993)		
Vendor	**Device Description**	**Available From/As of**
Panasonic	Panasonic KX-P1180 in Epson FX-86e mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P1191 in Epson FX-86e mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P1624 in Epson LQ-2500 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P1654 in Epson LQ-1050 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P1695 in Epson FX-1050 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P2123 in Epson LQ-860 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P2124 in Epson LQ-860 mode	OS/2 2.1 - (EPSON.DRV)
Panasonic	Panasonic KX-P2180 in Epson LQ-850 mode	OS/2 2.1 - (EPSON.DRV)
Panasonic	Panasonic KX-P2624 in Epson LQ-1050 mode	OS/2 2.0 - (EPSON.DRV)
Panasonic	Panasonic KX-P4410	OS/2 2.0 - (LASERJET.DRV)
Panasonic	Panasonic KX-P4420	OS/2 2.0 - (LASERJET.DRV)
Panasonic	Panasonic KX-P4430	OS/2 2.1 - (LASERJET.DRV)
Panasonic	Panasonic KX-P4450	OS/2 2.0 - (LASERJET.DRV)
Panasonic	Panasonic KX-P4450i	OS/2 2.0 - (LASERJET.DRV)
Panasonic	Panasonic KX-P4455 v51_4	OS/2 2.0 - (PSCRIPT.DRV)
Postscript (Generic)	Postscript Printer (not listed)	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS 860 Print System	OS/2 2.1 - (PSCRIPT.DRV)
QMS	QMS ColorScript 100	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS ColorScript 100 Mod 10	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS ColorScript 100 Mod 30	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS ColorScript 100 Mod 30si	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS IS X320T	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 410	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 800	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 800 Plus	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 810	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 810 Turbo	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 815	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 815 MR	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 820 Turbo	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 820	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 825	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 825 MR	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 1500	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 1700	OS/2 2.1 - (PSCRIPT.DRV)
QMS	QMS-PS 2000	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 2200	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 2210	OS/2 2.0 - (PSCRIPT.DRV)
QMS	QMS-PS 2220	OS/2 2.0 - (PSCRIPT.DRV)
Qume	Qume ScripTEN	OS/2 2.0 - (PSCRIPT.DRV)
Seiko	Seiko ColorPoint PS Model 04	OS/2 2.0 - (PSCRIPT.DRV)
Seiko	Seiko ColorPoint PS Model 14	OS/2 2.0 - (PSCRIPT.DRV)
Seiko	Seiko Personal ColorPoint PS	OS/2 2.0 - (PSCRIPT.DRV)
Seiko	Seiko Personal ColorPoint PSE	OS/2 2.1 - (PSCRIPT.DRV)
Star	Star NX-1000 Printer	OS/2 2.0 - (EPSON.DRV)
TI	TI 2115 (13 fonts) v47_0	OS/2 2.0 - (PSCRIPT.DRV)
TI	TI OmniLaser 2108	OS/2 2.0 - (PSCRIPT.DRV)
TI	TI Omnilaser 2115	OS/2 2.0 - (PSCRIPT.DRV)

Table 51 (Page 9 of 9). Printers and Plotters (as of May 10th 1993)		
Vendor	**Device Description**	**Available From/As of**
TI	TI microLaser PS17 v_52_1	OS/2 2.0 - (PSCRIPT.DRV)
TI	TI microLaser PS35 v_52_1	OS/2 2.0 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser 200e 17 fonts	OS/2 2.1 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser 200e 39 fonts	OS/2 2.1 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser 200i v2001_108	OS/2 2.1 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser II PX v2_02	OS/2 2.0 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser II PXi v2010	OS/2 2.0 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser II PXe 17 fonts	OS/2 2.1 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser II PXe 39 fonts	OS/2 2.1 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser IISD v2011	OS/2 2.0 - (PSCRIPT.DRV)
Tektronix	Tektronix Phaser III PXi v2010	OS/2 2.0 - (PSCRIPT.DRV)
Tektronix	Tektonix Phaser Card v1_1	OS/2 2.0 - (PSCRIPT.DRV)
Varityper	Varityper VT-600	OS/2 2.0 - (PSCRIPT.DRV)
Wang	Wang LCS15 FontPlus	OS/2 2.0 - (PSCRIPT.DRV)
Wang	Wang LCS15	OS/2 2.0 - (PSCRIPT.DRV)

C.11 LAN Adapter Support

The following table does not imply testing or certification by either IBM and/or National Software Testing Labs. For further clarification of testing or certification please contact either your dealer or the appropriate hardware manufacturer.

Table 52 (Page 1 of 4). LAN Adapter Support (as of May 10th 1993)		
Vendor	**Product Name**	**Available From**
3 Com Corp.	Etherlink II Model 3C503	LAN Server 3.0 & NTS/2 (ELNKII.OS/2)
3 Com Corp.	Etherlink II-16 Model 3C503-16	LAN Server 3.0 & NTS/2 (ELNKII.OS/2)
3 Com Corp.	Etherlink 16 Model 3C507	LAN Server 3.0 & NTS/2 (ELNK16.OS/2)
3 Com Corp.	Etherlink/MC Model 3C523	LAN Server 3.0 & NTS/2 (ELNKMC.OS/2)
3 Com Corp.	Etherlink/MC/32 Model 3C527	LAN Server 3.0 & NTS/2 (ELMC32.OS/2)
3 Com Corp.	Etherlink 3 Model 3C509 (ISA)	LAN Server 3.0 & NTS/2 (ELNK3.OS/2)
3 Com Corp.	Etherlink 3 Model 3C529 (MCA)	LAN Server 3.0 & NTS/2 (ELNK3.OS/2)
3 Com Corp.	Etherlink 3 Model 3C579 (EISA)	LAN Server 3.0 & NTS/2 (ELNK3.OS/2)
3 Com Corp.	Token ring 3 Model 3C609 (ISA)	LAN Server 3.0 & NTS/2 (IBMTOK.OS/2)
3 Com Corp.	Token ring 3 Model 3C629 (MCA)	LAN Server 3.0 & NTS/2 (IBMTOK.OS/2)
3 Com Corp.	Token ring 3 Model 3C679 (EISA)	LAN Server 3.0 & NTS/2 (IBMTOK.OS/2)
3 Com Corp.	FDDI-Link-STP Model 3C770 (EISA)	LAN Server 3.0 & NTS/2 (3C770.OS/2)
3 Com Corp.	FDDI-Link-F Model 3C771 (EISA)	LAN Server 3.0 & NTS/2 (3C770.OS/2)
Compaq	Netflex (32-bit dual speed TRN controller card)	LAN Server 3.0 & NTS/2
DCA	IRMAtrac - (NDIS 2.0 TRN - 16/4, ISA & MAC switchable)	LAN Server 3.0 & NTS/2
Digital Equipment Corporation	DE100 (Supports NDIS 2.01 for EISA-32bit, MCA and ATbus)	Digital
Digital Equipment Corporation	DE200 (Supports NDIS 2.01 for EISA-32bit, MCA and ATbus)	Digital
Digital Equipment Corporation	DE210 (Supports NDIS 2.01 for EISA-32bit, MCA and ATbus)	Digital
Digital Equipment Corporation	DEPCA (Supports NDIS 2.01 for EISA-32bit, MCA and ATbus)	Digital
Dowty Network Systems A/S	Dowty Network ScaNet	LAN Server 3.0 & NTS/2

Table 52 (Page 2 of 4). LAN Adapter Support (as of May 10th 1993)		
Vendor	**Product Name**	**Available From**
Fujitsu-ICL Systems, Inc.	Proprietry HDLC linkd	Fujitsu-ICL Systems, Inc.
G&H Consulting	Arcnet chip card	LAN Server 3.0 & NTS/2
IBM Corp.	FDDI Base Adapter/A Card	LAN Server 3.0 & NTS/2 Driver (IBMFDDI.OS/2)
IBM Corp.	FDDI Extender Adapter/A Card	LAN Server 3.0 & NTS/2 Driver (IBMFDDI.OS/2)
IBM Corp.	FDDI Copper Extender Adapter/A Card	LAN Server 3.0 & NTS/2 Driver (IBMFDDI.OS/2)
IBM Corp.	FDDI For Novell	LAN Server 3.0 & NTS/2 Driver (IBMFDDI0.OS/2)
IBM Corp.	Token Ring Network 16/4 Adapter/II Card	LAN Server 3.0 & NTS/2 Driver (IBM16TR.OS/2)
IBM Corp.	Ethernet Adapter/A Card	LAN Server 3.0 & NTS/2 Driver (MACETH.OS/2)
IBM Corp.	3174 Peer Communications	LAN Server 3.0 & NTS/2 Driver (IBM XLN.OS/2)
IBM Corp.	Token Ring Network Adapter/II Card	LAN Server 3.0 & NTS/2 Driver (IBMTOK.OS/2)
IBM Corp.	Token Ring 16/4 Adapter Card	LAN Server 3.0 & NTS/2 Driver (IBMTOK.OS/2)
IBM Corp.	Token Ring Network Adapter/IIA Card	LAN Server 3.0 & NTS/2 Driver (IBMTOK.OS/2)
IBM Corp.	Token Ring 16/4 Adapter/A Card	LAN Server 3.0 & NTS/2 Driver (IBMTOK.OS/2)
IBM Corp.	Token Ring Network PS/2 Model P70 386 Adapter Card	LAN Server 3.0 & NTS/2 Driver (IBMTOK.OS/2)
IBM Corp.	Token Ring 16/4 Busmaster Server Adapter Card	LAN Server 3.0 & NTS/2 Driver (IBMTRBM.OS/2)
IBM Corp.	PC Network Broadband Adapter II Card	LAN Server 3.0 & NTS/2 Driver (IBMNET.OS/2)
IBM Corp.	PC Network Adapter Card II Frequency 2	LAN Server 3.0 & NTS/2 Driver (IBMNET.OS/2)
IBM Corp.	PC Network Adapter Card II Frequency 3	LAN Server 3.0 & NTS/2 Driver (IBMNET.OS/2)
IBM Corp.	PC Network Baseband Adapter II Card	LAN Server 3.0 & NTS/2 Driver (IBMNET.OS/2)
IBM Corp.	PC Network Broadband Adapter II/A Card	LAN Server 3.0 & NTS/2 Driver (IBMNETA.OS/2)
IBM Corp.	PC Network Adapter Card II/A Frequency 2	LAN Server 3.0 & NTS/2 Driver (IBMNETA.OS/2)
IBM Corp.	PC Network Adapter Card II/A Frequency 3	LAN Server 3.0 & NTS/2 Driver (IBMNETA.OS/2)
IBM Corp.	PC Network Baseband Adapter II/A Card	LAN Server 3.0 & NTS/2 Driver (IBMNETA.OS/2)
IBM Corp.	LAN Adapter for Ethernet	LAN Server 3.0 & NTS/2 Driver (IBMENI.OS/2)
IBM Corp.	LAN Adapter/A for Ethernet	LAN Server 3.0 & NTS/2 Driver (IBMENIA.OS/2)
Intel Corp.	Ethernet Express 16 - Ethernet Card	
Intel Corp.	Ethernet Express 32 - Ethernet Card	LAN Server 3.0 & NTS/2
Intel Corp.	Ethernet Express MCA - Ethernet Card	LAN Server 3.0 & NTS/2
Intel Corp.	593 Ethernet NDIS Driver	Intel
Madge Networks, Inc.	TRN Adapter support	Madge Networks, Inc
National Semiconductor	Ethernode - 16AT 4-5 Models	National Semiconductor
NCR Systems Engineering	WaveLan - wireless Ethernet	LAN Server 3.0 & NTS/2
NCR Systems Engineering	WaveLan - AT/MA	LAN Server 3.0 & NTS/2

Table 52 (Page 3 of 4). LAN Adapter Support (as of May 10th 1993)

Vendor	Product Name	Available From
NEC Information Systems, Inc.	NEC Netcard	NEC Information Systems, Inc.
Network Controls International	NCI BANCRING	Network Controls International
SMC Corp.	TokenCard Elite SMC-TRE (16-bit AT EISA)(8-bit XT)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	TokenCard Elite/A SMC-TRE/A (16-bit MCA)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 16T (16-bit-AT, 8-bit-XT)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 10T (8-bit-XT)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 10T/A (16-bit-MCA)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 32T (32-bit-EISA)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 16 COMB (16-bit-AT, 8-bit-XT)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 16 (16-bit-AT, 8-bit-XT)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite (8-bit-XT)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite/A (16-bit-MCA)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
SMC Corp.	EtherCard PLUS Elite 32 (32-bit-EISA)	LAN Server 3.0 & NTS/2 Drivers (SMCMAC.OS/2 or SCMMAC2.OS/2)
Tiara Computer Systems, Inc.	Ethernet Universal LanCard E2000	Tiara Computer Systems, Inc.
Tiara Computer Systems, Inc.	Arcnet Universal Lan Card. TM	Tiara Computer Systems, Inc.
Tiara Computer Systems, Inc.	10NET	Tiara Computer Systems, Inc.
Thomas Conrad	TRN - TC4945	Thomas Conrad
Thomas Conrad	ARCNET - TC6245	Thomas Conrad
Ungerman-Bass	NIU-PCEOTP	LAN Server 3.0 & NTS/2
Ungerman-Bass	Ethernext	LAN Server 3.0 & NTS/2
Ungerman-Bass	Access	LAN Server 3.0 & NTS/2
Ungerman-Bass	Bass NUI pc Model PC2030	LAN Server 3.0 & NTS/2 (UBNEI OS/2)
Ungerman-Bass	Bass NUI ps/2 Model PC2030	LAN Server 3.0 & NTS/2 (UBNEI.OS/2)
Western Digital	Ethercard Plus	LAN Server 3.0 & NTS/2 (MACWD.OS/20
Western Digital	Ethercard Plus/A	LAN Server 3.0 & NTS/2 (MACWD.OS/20
Zenith	LAN10 series-Ethernet	Zenith - (MACZEC OS/2)/(MACZEC2.OS2) Zenith Tech Support - (708) 391-8901
Zenith	LAN10E (8-bit, 16-bit BNC.AUI)	Zenith - (MACZEC OS/2)/(MACZEC2.OS2) Zenith Tech Support - (708) 391-8901
Zenith	LAN10T (16-bit, 8-bit Drivers)	Zenith - (MACZEC OS/2)/(MACZEC2.OS2) Zenith Tech Support - (708) 391-8901
Zenith	LAN10M (16-bit, 8-bit Drivers)	Zenith - (MACZEC OS/2)/(MACZEC2.OS2) Zenith Tech Support - (708) 391-8901
Zenith	LAN16 TR series-TRN (16/4 TRN)	Zenith - (ZEN16NDS.OS/2) Zenith Tech Support - (708) 391-8901
Zenith	LAN16 TRF series - (16/4 TRN)	Zenith - (ZEN16NDS.OS/2) Zenith Tech Support - (708) 391-8901

Table 52 (Page 4 of 4). LAN Adapter Support (as of May 10th 1993)		
Vendor	**Product Name**	**Available From**
Zenith	LAN4TR 4 Mbps TRN (8-bit, 16-bit MCA Drivers)	Zenith - (ZEN4LM.OS/2) Zenith Tech Support - (708) 391-8901
Zenith	LAN4000C - (8-bit MCA)	Zenith - (MACZLAN.OS/2) Zenith Tech Support - (708) 391-8901
Zenith Data Systems - Groupe Bull	Ethernet integrated network controller	Zenith

C.12 Tape Drives

The following table shows tape hardware devices that are supported under OS/2 by third parties. IBM makes no warranty, express or implied, with respect to the operation of these devices with the software listed below nor to the completeness of this list. When in doubt you should seek additional information from your dealer or the appropriate vendor.

Table 53 (Page 1 of 3). Tape Support (as of May 10th 1993)		
Tape Hardware Vendor	**Tape Product Name**	**Software Support Available From**
ADIC	ADIC DAT 1300	NovaBack for OS/2
ADIC	ADIC DAT 2000a	NovaBack for OS/2
ADIC	ADIC DAT 8000a	NovaBack for OS/2
Alliance	QA60/QA150/QA1200	Sytos Plus Ver 1.35
Archive	Archive 2060S	NovaBack for OS/2
Archive	Archive 2150S	NovaBack for OS/2
Archive	Archive 2525S	NovaBack for OS/2
Archive	Archive Anaconda	NovaBack for OS/2
Archive	Archive Python DAT	NovaBack for OS/2
Archive	Archive Python DDS-DC	NovaBack for OS/2
Archive	Archive Python 4320/4520 DAT	Sytos Plus Ver 1.37
Cipher	Cipher ST150	NovaBack for OS/2
Cipher	Cipher 5400/5400 Plus	Sytron Plus Ver 1.35
Cipher	Cipher Ciera	NovaBack for OS/2
Cipher	Cipher F880ES	NovaBack for OS/2
Cipher	Cipher C995S	NovaBack for OS/2
Cipher	Cipher T480	NovaBack for OS/2
Colorado Memory Systems	CMS Jumbo Tape Drives (120, 250)	Sytos Plus (V1.36) for OS/2 for Jumbo Tape Drives, Available from CMS
Colorado Memory Systems	CMS Power Tape	Sytos Plus (V1.36) for OS/2 for PowerTape, Available from CMS
Compaq	Compaq 150/250	Sytos Plus Ver 1.35
Dynatek	Dynatek DAT 2000	NovaBack for OS/2
Dynatek	Dynatek DAT 2600	NovaBack for OS/2
Dynatek	Dynatek DAT 4000	NovaBack for OS/2
Dynatek	Dynatek HSB 2300	NovaBack for OS/2
Compaq	Compaq 320/525	Sytos Plus Ver 1.35
Compaq	Compaq 1.3/2.0 DAT	Sytos Plus Ver 1.35
Exabyte	Exabyte EXB-8200	Sytos Plus Ver 1.37, NovaBack for OS/2
ExaByte	ExaByte EXB 8200SX	NovaBack for OS/2
ExaByte	ExaByte 8500	NovaBack for OS/2
ExaByte	ExaByte 10i	NovaBack for OS/2

Table 53 (Page 2 of 3). Tape Support (as of May 10th 1993)

Tape Hardware Vendor	Tape Product Name	Software Support Available From
Fujitsu	Fujitsu M2481A	NovaBack for OS/2
Fujitsu	Fujitsu 2483	NovaBack for OS/2
Hewlett Packard	Hewlett Packard 35470A 4mm DAT	Sytos Plus Ver 1.37, NovaBack for OS/2
Hewlett Packard	Hewlett Packard 35480 4mm DAT	Sytos Plus Ver 1.37, NovaBack for OS/2
Hewlett Packard	Hewlett Packard 88780 9 Track	Sytos Plus Ver 1.37, NovaBack for OS/2
IBM	IBM 1.2GB External 001	Sytos Plus Ver 1.37
IBM	IBM 5.0GB Tape 001	Sytos Plus Ver 1.37
IBM	IBM PS/2 3.5" Optical Drive	Sytos Plus Ver 1.37
IBM	IBM 2.0GB Tape Drive	Sytos Plus Ver 1.37
IBM	IBM 2.3GB Tape Drive	Sytos Plus Ver 1.37
IBM	IBM 6157 Streaming Tape Drive	Sytos Plus Ver 1.37
IBM	IBM PS/2 2.88MB Diskette Drive	Sytos Plus Ver 1.37
Identica	IDT-P60/250	NovaBack for OS/2
Identica	IDT-150	Sytos Plus Ver 1.35
Identica	IDT-150/250	NovaBack for OS/2
Identica	IDT-320/525	NovaBack for OS/2
Identica	IDT-1000	NovaBack for OS/2
Identica	IDT-2000	NovaBack for OS/2
Legacy	Legacy 45/60	Sytos Plus Ver 1.37
Legacy	Legacy 150S/X	Sytos Plus Ver 1.37
Legacy	Legacy 500s	Sytos Plus Ver 1.37
Legacy	Legacy 1000s	Sytos Plus Ver 1.37
Legacy	Legacy 2000s	Sytos Plus Ver 1.37
LMSI	LMSI 9490	NovaBack for OS/2
LMSI	LMSI 1545	NovaBack for OS/2
LMSI	LMSI Keystone	NovaBack for OS/2
M4 Data	M4 9905 9 Track	NovaBack for OS/2
M4 Data	M4 9914 9 Track	NovaBack for OS/2
Parastor	Parastor 250	NovaBack for OS/2
Parastor	Parastor 525	NovaBack for OS/2
Parastor	Parastor 1000	NovaBack for OS/2
Qualstar	Qualstar 1054	NovaBack for OS/2
Qualstar	Qualstar 1260	NovaBack for OS/2
Qualstar	Qualstar 3412	NovaBack for OS/2
R-Byte	r-100	NovaBack for OS/2
Sankyo	CP150	NovaBack for OS/2
Sankyo	CP320	NovaBack for OS/2
Soltronics	David I	NovaBack for OS/2
Soltronics	David II	NovaBack for OS/2
Soltronics	Samson II	NovaBack for OS/2
Sony	SDT1000	NovaBack for OS/2
Sony	SDT2000	NovaBack for OS/2
StorageTek	StorageTek 4220	NovaBack for OS/2
StorageTek	StorageTek 4280	NovaBack for OS/2
StorageTek	StorageTek 9914	NovaBack for OS/2
Storage Solutions	SSI DATa-bak	NovaBack for OS/2
Tandberg	Tandberg 3610	Sytos Plus Ver 1.35
Tandberg	Tandberg 3650	Sytos Plus Ver 1.35

Table 53 (Page 3 of 3). Tape Support (as of May 10th 1993)

Tape Hardware Vendor	Tape Product Name	Software Support Available From
Tandberg	Tandberg 3660	NovaBack for OS/2
Tandberg	Tandberg 3820	Sytos Plus Ver 1.37, NovaBack for OS/2
Tandberg	Tandberg 4100	NovaBack for OS/2
Tandberg	Tandberg 4120	Sytos Plus Ver 1.37
Teac	MT-01N	NovaBack for OS/2
Tecmar	Tecmar QT-60	Sytos Plus Ver 1.35
Tecmar	Tecmar QT-100	Sytos Plus Ver 1.35
Tecmar	Tecmar QT-125	Sytos Plus Ver 1.35
Tecmar	Tecmar QT-250/QT-250ES/Proline 250	Sytos Plus Ver 1.37
Tecmar	Tecmar QT-525ES/Proline 525ES	Sytos Plus Ver 1.37
Tecmar	Tecmar QT 1000ES/Proline 1000	Sytos Plus Ver 1.37
Tecmar	Tecmar THS-2200	Sytos Plus Ver 1.37
Tecmar	Tecmar DataVault	Sytos Plus Ver 1.37
Tecmar	Tecmar Proline DAT	Sytos Plus Ver 1.37
Tecmar	Tecmar Proline 2200	Sytos Plus Ver 1.37
Tecmar	Tecmar 4000 DAT	Sytos Plus Ver 1.37
Wangdat	Wangdat 1300	Sytos Plus Ver 1.37, NovaBack for OS/2
Wangdat	Wangdat 1300XL	NovaBack for OS/2
Wangdat	Wangdat 2000	NovaBack for OS/2
Wangdat	Wangdat 2600	NovaBack for OS/2
Wangdat	Wangdat 3200	Sytos Plus Ver 1.37, NovaBack for OS/2
Wangtek	Wangtek 5099	NovaBack for OS/2
Wangtek	Wangtek 5100	NovaBack for OS/2
Wangtek	Wangtek 5150ES	Sytos Plus Ver 1.37, NovaBack for OS/2
Wangtek	Wangtek 5525ES	Sytos Plus Ver 1.37, NovaBack for OS/2
Wangtek	Wangtek 51000ES	Sytos Plus Ver 1.37
Wangtek	Wangtek 6130/6200 DAT	Sytos Plus Ver 1.37, NovaBack for OS/2
Wangtek	Wangtek 7200	NovaBack for OS/2
Wangtek	Wangtek PC-36 60MB	Sytos Plus Ver 1.35
Wangtek	Wangtek PC-36 125MB	Sytos Plus Ver 1.35
Wangtek	Wangtek PC-36 150MB	Sytos Plus Ver 1.35
Wangtek	Wangtek PC-02 150/250MB	Sytos Plus Ver 1.35
Wangtek	Wangtek MC-02 150/250MB	Sytos Plus Ver 1.35

Appendix D. APAR Fixes in OS/2 2.00.1, Service Pak XR06055, and OS/2 2.1

This list of APAR fixes in OS/2 2.00.1, Service Pak XR06055 and OS/2 2.1 has been included for quick reference purposes, and is an abbreviated version of the lists of APAR fixes provided by the IBM OS/2 Service organization in the Personal Systems Programming laboratory at Boca Raton, Florida.

For a more detailed description of each fix, please contact the IBM Service organization in your country,

Table 54 (Page 1 of 19). APAR Fixes in OS/2 2.x

APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
JR05252			Y
JR05522	Y		Y
JR05700			Y
JR05775	Y		Y
JR05855	Y		Y
JR05876			Y
JR06197			Y
JR06297			Y
JR06336			Y
JR06367			Y
JR06389			Y
JR06471			Y
JR06685			Y
PJ01512	Y	Y	Y
PJ01667	Y	Y	Y
PJ01696		Y	Y
PJ01861	Y	Y	Y
PJ02058	Y	Y	Y
PJ02271	Y	Y	Y
PJ02977	Y	Y	Y
PJ03080	Y	Y	Y
PJ03143	Y		Y
PJ03174	Y		Y
PJ03233	Y	Y	Y
PJ03246		Y	Y
PJ03268	Y	Y	Y
PJ03466			Y
PJ03499	Y	Y	Y
PJ03510	Y	Y	Y
PJ03534	Y	Y	Y
PJ03552			Y
PJ03560			Y
PJ03582	Y		Y
PJ03606	Y	Y	Y
PJ03614			Y
PJ03638		Y	Y

Table 54 (Page 2 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ03645			Y
PJ03647		Y	Y
PJ03652	Y		Y
PJ03656	Y	Y	Y
PJ03661	Y	Y	Y
PJ03666		Y	Y
PJ03668		Y	Y
PJ03671	Y		Y
PJ03677	Y	Y	Y
PJ03684	Y		Y
PJ03685	Y		Y
PJ03686	Y	Y	Y
PJ03690			Y
PJ03693			Y
PJ03696	Y	Y	Y
PJ03700	Y	Y	Y
PJ03701		Y	Y
PJ03705	Y	Y	Y
PJ03707			Y
PJ03708		Y	Y
PJ03709	Y	Y	Y
PJ03716		Y	Y
PJ03717	Y	Y	Y
PJ03719			Y
PJ03720			Y
PJ03721	Y	Y	Y
PJ03726	Y	Y	Y
PJ03733	Y	Y	Y
PJ03738	Y		Y
PJ03739	Y	Y	Y
PJ03740	Y	Y	Y
PJ03743		Y	Y
PJ03748	Y	Y	Y
PJ03755		Y	Y
PJ03756	Y	Y	Y
PJ03757		Y	Y
PJ03763			Y
PJ03768			Y
PJ03771		Y	Y
PJ03780		Y	Y
PJ03783	Y	Y	Y
PJ03785		Y	Y
PJ03788	Y		Y
PJ03803		Y	Y
PJ03807	Y	Y	Y
PJ03810	Y	Y	Y
PJ03811		Y	Y
PJ03812		Y	Y

Table 54 (Page 3 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ03823	Y	Y	Y
PJ03828			Y
PJ03833	Y	Y	Y
PJ03836	Y	Y	Y
PJ03837	Y	Y	Y
PJ03838		Y	Y
PJ03849	Y		Y
PJ03852		Y	Y
PJ03854	Y	Y	Y
PJ03863	Y	Y	Y
PJ03865			Y
PJ03875		Y	Y
PJ03889	Y	Y	Y
PJ03893			Y
PJ03897	Y	Y	Y
PJ03901	Y	Y	Y
PJ03904		Y	Y
PJ03905	Y	Y	Y
PJ03906		Y	Y
PJ03911	Y		Y
PJ03912		Y	Y
PJ03918	Y	Y	Y
PJ03925			Y
PJ03928	Y	Y	Y
PJ03932			Y
PJ03933		Y	Y
PJ03937	Y	Y	Y
PJ03939	Y		Y
PJ03942		Y	Y
PJ03945	Y	Y	Y
PJ03946	Y	Y	Y
PJ03956		Y	Y
PJ03960		Y	Y
PJ03970	Y	Y	Y
PJ03973	Y	Y	Y
PJ03974		Y	Y
PJ03981		Y	Y
PJ03997	Y		Y
PJ03998	Y	Y	Y
PJ04000		Y	Y
PJ04006	Y	Y	Y
PJ04018	Y	Y	Y
PJ04023	Y	Y	Y
PJ04025	Y	Y	Y
PJ04026		Y	Y
PJ04027	Y	Y	Y
PJ04032		Y	Y
PJ04036		Y	Y

Table 54 (Page 4 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ04045		Y	Y
PJ04046			Y
PJ04050	Y	Y	Y
PJ04052	Y	Y	Y
PJ04054	Y	Y	Y
PJ04056	Y	Y	Y
PJ04060	Y		Y
PJ04063			Y
PJ04064		Y	Y
PJ04067			Y
PJ04068	Y		Y
PJ04070	Y		Y
PJ04072	Y		Y
PJ04076		Y	Y
PJ04081	Y		Y
PJ04082	Y	Y	Y
PJ04090	Y		Y
PJ04092			Y
PJ04099	Y	Y	Y
PJ04100			Y
PJ04106			Y
PJ04108			Y
PJ04108			Y
PJ04122			Y
PJ04125	Y	Y	Y
PJ04128		Y	Y
PJ04133	Y	Y	Y
PJ04136	Y	Y	Y
PJ04139	Y		Y
PJ04143			Y
PJ04150	Y	Y	Y
PJ04155	Y	Y	Y
PJ04157			Y
PJ04159	Y	Y	Y
PJ04164	Y		Y
PJ04170		Y	Y
PJ04180			Y
PJ04181		Y	Y
PJ04191	Y		Y
PJ04204	Y	Y	Y
PJ04207	Y		Y
PJ04208	Y	Y	Y
PJ04211		Y	Y
PJ04212		Y	Y
PJ04213		Y	Y
PJ04215		Y	Y
PJ04217	Y	Y	Y
PJ04220		Y	Y

Table 54 (Page 5 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ04222	Y	Y	Y
PJ04240	Y		Y
PJ04241			Y
PJ04242		Y	Y
PJ04248	Y	Y	Y
PJ04253		Y	Y
PJ04254		Y	Y
PJ04256	Y	Y	Y
PJ04258			Y
PJ04264		Y	Y
PJ04265	Y		Y
PJ04266		Y	Y
PJ04267	Y	Y	Y
PJ04268	Y	Y	Y
PJ04273	Y	Y	Y
PJ04276	Y		Y
PJ04284		Y	Y
PJ04287	Y	Y	Y
PJ04288			Y
PJ04289			Y
PJ04290		Y	Y
PJ04292	Y	Y	Y
PJ04296			Y
PJ04300			Y
PJ04303	Y	Y	Y
PJ04305	Y	Y	Y
PJ04308	Y	Y	Y
PJ04313		Y	Y
PJ04318			Y
PJ04322			Y
PJ04324		Y	Y
PJ04325		Y	Y
PJ04326	Y	Y	Y
PJ04327			Y
PJ04337		Y	Y
PJ04338	Y	Y	Y
PJ04344		Y	Y
PJ04348		Y	Y
PJ04358		Y	Y
PJ04366	Y	Y	Y
PJ04370	Y	Y	Y
PJ04374			Y
PJ04377	Y	Y	Y
PJ04380	Y	Y	Y
PJ04402		Y	Y
PJ04405			Y
PJ04417	Y	Y	Y
PJ04419			Y

Table 54 (Page 6 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ04432			Y
PJ04434			Y
PJ04440		Y	Y
PJ04444	Y	Y	Y
PJ04446			Y
PJ04456		Y	Y
PJ04463			Y
PJ04470		Y	Y
PJ04482	Y	Y	Y
PJ04489	Y	Y	Y
PJ04494	Y	Y	Y
PJ04499		Y	Y
PJ04500	Y	Y	Y
PJ04502	Y	Y	Y
PJ04507	Y	Y	Y
PJ04508			Y
PJ04509		Y	Y
PJ04515	Y	Y	Y
PJ04519	Y	Y	Y
PJ04524			Y
PJ04527			Y
PJ04539			Y
PJ04540		Y	Y
PJ04548	Y	Y	Y
PJ04553	Y	Y	Y
PJ04564		Y	Y
PJ04567	Y	Y	Y
PJ04572		Y	Y
PJ04573			Y
PJ04583	Y	Y	Y
PJ04587		Y	Y
PJ04596	Y	Y	Y
PJ04600		Y	Y
PJ04615		Y	Y
PJ04628	Y	Y	Y
PJ04640	Y	Y	Y
PJ04662		Y	Y
PJ04661		Y	Y
PJ04664		Y	Y
PJ04669		Y	Y
PJ04676			Y
PJ04680		Y	Y
PJ04691			Y
PJ04695	Y	Y	Y
PJ04696		Y	Y
PJ04699	Y	Y	Y
PJ04706		Y	Y
PJ04711	Y	Y	Y

Table 54 (Page 7 of 19). APAR Fixes in OS/2 2.x

APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ04713		Y	Y
PJ04727	Y	Y	Y
PJ04730	Y	Y	Y
PJ04736	Y	Y	Y
PJ04782			Y
PJ04791	Y	Y	Y
PJ04796			Y
PJ04804			Y
PJ04808	Y		Y
PJ04818			Y
PJ04819		Y	Y
PJ04835		Y	Y
PJ04836		Y	Y
PJ04837			Y
PJ04845	Y	Y	Y
PJ04848			Y
PJ04854	Y	Y	Y
PJ04876			Y
PJ04878		Y	Y
PJ04882		Y	Y
PJ04883	Y	Y	Y
PJ04886			Y
PJ04890		Y	Y
PJ04895		Y	Y
PJ04897	Y	Y	Y
PJ04904		Y	Y
PJ04912			Y
PJ04916			Y
PJ04920		Y	Y
PJ04926	Y	Y	Y
PJ04930		Y	Y
PJ04937			Y
PJ04941	Y	Y	Y
PJ04942		Y	Y
PJ04945	Y	Y	Y
PJ04960		Y	Y
PJ04963			Y
PJ04964	Y	Y	Y
PJ04967	Y	Y	Y
PJ04968		Y	Y
PJ04971			Y
PJ04983	Y	Y	Y
PJ04993	Y	Y	Y
PJ04999	Y	Y	Y
PJ05004			Y
PJ05012	Y	Y	Y
PJ05032		Y	Y
PJ05039			Y

Table 54 (Page 8 of 19). APAR Fixes in OS/2 2.x

APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ05044		Y	Y
PJ05052		Y	Y
PJ05061	Y	Y	Y
PJ05064			Y
PJ05075	Y	Y	Y
PJ05079	Y	Y	Y
PJ05090		Y	Y
PJ05092			Y
PJ05093		Y	Y
PJ05097		Y	Y
PJ05098	Y	Y	Y
PJ05121			Y
PJ05147	Y	Y	Y
PJ05151	Y	Y	Y
PJ05157		Y	Y
PJ05160		Y	Y
PJ05162		Y	Y
PJ05175			Y
PJ05179		Y	Y
PJ05183			Y
PJ05185		Y	Y
PJ05187	Y	Y	Y
PJ05189	Y	Y	Y
PJ05191		Y	Y
PJ05193			Y
PJ05206			Y
PJ05212	Y	Y	Y
PJ05214		Y	Y
PJ05215		Y	Y
PJ05226		Y	Y
PJ05232		Y	Y
PJ05235			Y
PJ05245			Y
PJ05247		Y	Y
PJ05251			Y
PJ05253			Y
PJ05257	Y	Y	Y
PJ05268	Y	Y	Y
PJ05290		Y	Y
PJ05292		Y	Y
PJ05295			Y
PJ05309			Y
PJ05310		Y	Y
PJ05318		Y	Y
PJ05332			Y
PJ05333			Y
PJ05354		Y	Y
PJ05360	Y	Y	Y

Table 54 (Page 9 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ05370		Y	Y
PJ05374			Y
PJ05376			Y
PJ05377	Y	Y	Y
PJ05378	Y	Y	Y
PJ05388			Y
PJ05390			Y
PJ05395			Y
PJ05398			Y
PJ05401			Y
PJ05404		Y	Y
PJ05408			Y
PJ05411		Y	Y
PJ05412			Y
PJ05414			Y
PJ05415			Y
PJ05419	Y	Y	Y
PJ05421			Y
PJ05433		Y	Y
PJ05472	Y	Y	Y
PJ05476		Y	Y
PJ05480		Y	Y
PJ05491	Y	Y	Y
PJ05497	Y	Y	Y
PJ05506		Y	Y
PJ05511		Y	Y
PJ05517			Y
PJ05519			Y
PJ05528			Y
PJ05533	Y	Y	Y
PJ05541			Y
PJ05545			Y
PJ05546			Y
PJ05555			Y
PJ05555			Y
PJ05561			Y
PJ05562			Y
PJ05572			Y
PJ05573	Y	Y	Y
PJ05574	Y	Y	Y
PJ05583			Y
PJ05594		Y	Y
PJ05608			Y
PJ05613			Y
PJ05622			Y
PJ05628			Y
PJ05641			Y
PJ05643			Y

Table 54 (Page 10 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ05649			Y
PJ05652			Y
PJ05655		Y	Y
PJ05659			Y
PJ05673		Y	Y
PJ05674	Y	Y	Y
PJ05676			Y
PJ05679		Y	Y
PJ05682	Y	Y	Y
PJ05683		Y	Y
PJ05685			Y
PJ05710			Y
PJ05712			Y
PJ05720			Y
PJ05721			Y
PJ05727			Y
PJ05729			Y
PJ05731		Y	Y
PJ05742		Y	Y
PJ05745			Y
PJ05751			Y
PJ05753			Y
PJ05763		Y	Y
PJ05769		Y	Y
PJ05772			Y
PJ05778			Y
PJ05787		Y	Y
PJ05795			Y
PJ05813			Y
PJ05820			Y
PJ05823			Y
PJ05828			Y
PJ05834			Y
PJ05837			Y
PJ05842			Y
PJ05845			Y
PJ05849	Y	Y	Y
PJ05862			Y
PJ05865			Y
PJ05866			Y
PJ05870			Y
PJ05873			Y
PJ05880			Y
PJ05896			Y
PJ05897	Y	Y	Y
PJ05901			Y
PJ05906			Y
PJ05908	Y	Y	Y

Table 54 (Page 11 of 19). APAR Fixes in OS/2 2.x

APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ05920	Y	Y	Y
PJ05921			Y
PJ05926			Y
PJ05929			Y
PJ05941			Y
PJ05949			Y
PJ05950			Y
PJ05962			Y
PJ05968			Y
PJ05969			Y
PJ05974			Y
PJ05979			Y
PJ05985	Y	Y	Y
PJ05991			Y
PJ06004			Y
PJ06008			Y
PJ06014		Y	Y
PJ06015		Y	Y
PJ06016		Y	Y
PJ06017		Y	Y
PJ06018	Y	Y	Y
PJ06019	Y	Y	Y
PJ06025	Y	Y	Y
PJ06031	Y	Y	Y
PJ06032		Y	Y
PJ06035		Y	Y
PJ06037			Y
PJ06045	Y	Y	Y
PJ06050		Y	Y
PJ06052		Y	Y
PJ06053		Y	Y
PJ06055			Y
PJ06057	Y	Y	Y
PJ06058	Y	Y	Y
PJ06059	Y	Y	Y
PJ06060	Y	Y	Y
PJ06070			Y
PJ06071	Y	Y	Y
PJ06072	Y	Y	Y
PJ06073	Y	Y	Y
PJ06074	Y	Y	Y
PJ06075	Y	Y	Y
PJ06076	Y	Y	Y
PJ06077		Y	Y
PJ06078		Y	Y
PJ06079		Y	Y
PJ06080		Y	Y
PJ06081		Y	Y

Table 54 (Page 12 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ06083		Y	Y
PJ06084		Y	Y
PJ06087		Y	Y
PJ06088		Y	Y
PJ06089		Y	Y
PJ06093			Y
PJ06094		Y	Y
PJ06101		Y	Y
PJ06102		Y	Y
PJ06103		Y	Y
PJ06104		Y	Y
PJ06105		Y	Y
PJ06106		Y	Y
PJ06113		Y	Y
PJ06114		Y	Y
PJ06117		Y	Y
PJ06119		Y	Y
PJ06120			Y
PJ06126			Y
PJ06132			Y
PJ06136			Y
PJ06137	Y	Y	Y
PJ06138	Y	Y	Y
PJ06147			Y
PJ06149		Y	Y
PJ06150		Y	Y
PJ06151		Y	Y
PJ06152	Y	Y	Y
PJ06153	Y	Y	Y
PJ06154		Y	Y
PJ06166			Y
PJ06177			Y
PJ06179			Y
PJ06183			Y
PJ06189			Y
PJ06201			Y
PJ06202			Y
PJ06203			Y
PJ06204		Y	Y
PJ06217		Y	Y
PJ06227			Y
PJ06238			Y
PJ06250	Y	Y	Y
PJ06259			Y
PJ06260			Y
PJ06263		Y	Y
PJ06266			Y
PJ06274		Y	Y

Table 54 (Page 13 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ06275		Y	Y
PJ06276			Y
PJ06277		Y	Y
PJ06280		Y	Y
PJ06281		Y	Y
PJ06283		Y	Y
PJ06286		Y	Y
PJ06287			Y
PJ06288			Y
PJ06289			Y
PJ06290		Y	Y
PJ06291		Y	Y
PJ06294			Y
PJ06305			Y
PJ06323			Y
PJ06337			Y
PJ06339			Y
PJ06352			Y
PJ06357			Y
PJ06394			Y
PJ06401			Y
PJ06423			Y
PJ06424			Y
PJ06427			Y
PJ06428			Y
PJ06431			Y
PJ06435			Y
PJ06444			Y
PJ06445			Y
PJ06452			Y
PJ06456			Y
PJ06462			Y
PJ06463			Y
PJ06465			Y
PJ06494			Y
PJ06496			Y
PJ06497			Y
PJ06497			Y
PJ06497			Y
PJ06499			Y
PJ06504			Y
PJ06507			Y
PJ06509			Y
PJ06511			Y
PJ06511			Y
PJ06514			Y
PJ06515			Y
PJ06524			Y

Table 54 (Page 14 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ06524			Y
PJ06531			Y
PJ06533			Y
PJ06544			Y
PJ06545			Y
PJ06550			Y
PJ06558			Y
PJ06561			Y
PJ06566			Y
PJ06569			Y
PJ06571			Y
PJ06586			Y
PJ06599			Y
PJ06605			Y
PJ06607			Y
PJ06608			Y
PJ06608			Y
PJ06614			Y
PJ06619			Y
PJ06622			Y
PJ06629			Y
PJ06632			Y
PJ06638			Y
PJ06639			Y
PJ06640			Y
PJ06644			Y
PJ06658			Y
PJ06671			Y
PJ06673			Y
PJ06675			Y
PJ06677			Y
PJ06679			Y
PJ06684			Y
PJ06699			Y
PJ06706			Y
PJ06713			Y
PJ06716			Y
PJ06717			Y
PJ06717			Y
PJ06718			Y
PJ06719			Y
PJ06722			Y
PJ06724			Y
PJ06735			Y
PJ06736			Y
PJ06744			Y
PJ06782			Y
PJ06783			Y

Table 54 (Page 15 of 19). APAR Fixes in OS/2 2.x

APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ06786			Y
PJ06793			Y
PJ06797			Y
PJ06801			Y
PJ06801			Y
PJ06803			Y
PJ06804			Y
PJ06810			Y
PJ06811			Y
PJ06818			Y
PJ06819			Y
PJ06826			Y
PJ06830			Y
PJ06844			Y
PJ06847			Y
PJ06847			Y
PJ06848			Y
PJ06848			Y
PJ06852			Y
PJ06857			Y
PJ06863			Y
PJ06866			Y
PJ06867			Y
PJ06867			Y
PJ06870			Y
PJ06871			Y
PJ06880			Y
PJ06883			Y
PJ06884			Y
PJ06888			Y
PJ06889			Y
PJ06892			Y
PJ06894			Y
PJ06899			Y
PJ06906			Y
PJ06922			Y
PJ06923			Y
PJ06926			Y
PJ06930			Y
PJ06936			Y
PJ06938			Y
PJ06940			Y
PJ06941			Y
PJ06948			Y
PJ06949			Y
PJ06951			Y
PJ06959			Y
PJ06985			Y

Table 54 (Page 16 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ06992			Y
PJ06993			Y
PJ07002			Y
PJ07009			Y
PJ07013			Y
PJ07022			Y
PJ07023			Y
PJ07023			Y
PJ07027			Y
PJ07029			Y
PJ07050			Y
PJ07051			Y
PJ07054			Y
PJ07055			Y
PJ07079			Y
PJ07098			Y
PJ07104			Y
PJ07110			Y
PJ07120			Y
PJ07122			Y
PJ07122			Y
PJ07124			Y
PJ07126			Y
PJ07129			Y
PJ07131			Y
PJ07138			Y
PJ07139			Y
PJ07149			Y
PJ07157			Y
PJ07162			Y
PJ07163			Y
PJ07175			Y
PJ07180			Y
PJ07180			Y
PJ07180			Y
PJ07189			Y
PJ07196			Y
PJ07201			Y
PJ07202			Y
PJ07205			Y
PJ07214			Y
PJ07217			Y
PJ07222			Y
PJ07228			Y
PJ07231			Y
PJ07234			Y
PJ07235			Y
PJ07236			Y

Table 54 (Page 17 of 19). APAR Fixes in OS/2 2.x			
APAR	**OS/2 2.00.1**	**Service Pak XR06055**	**OS/2 2.1**
PJ07238			Y
PJ07239			Y
PJ07241			Y
PJ07246			Y
PJ07250			Y
PJ07251			Y
PJ07252			Y
PJ07255			Y
PJ07263			Y
PJ07268			Y
PJ07272			Y
PJ07274			Y
PJ07278			Y
PJ07281			Y
PJ07284			Y
PJ07285			Y
PJ07296			Y
PJ07301			Y
PJ07305			Y
PJ07314			Y
PJ07322			Y
PJ07329			Y
PJ07334			Y
PJ07354			Y
PJ07363			Y
PJ07391			Y
PJ07392			Y
PJ07394			Y
PJ07396			Y
PJ07398			Y
PJ07399			Y
PJ07410			Y
PJ07417			Y
PJ07425			Y
PJ07428			Y
PJ07432			Y
PJ07441			Y
PJ07442			Y
PJ07445			Y
PJ07447			Y
PJ07467			Y
PJ07468			Y
PJ07494			Y
PJ07511			Y
PJ07517			Y
PJ07525			Y
PJ07526			Y
PJ07532			Y

Table 54 (Page 18 of 19). APAR Fixes in OS/2 2.x			
APAR	**OS/2 2.00.1**	**Service Pak XR06055**	**OS/2 2.1**
PJ07538			Y
PJ07557			Y
PJ07569			Y
PJ07583			Y
PJ07586			Y
PJ07591			Y
PJ07598			Y
PJ07617			Y
PJ07619			Y
PJ07620			Y
PJ07620			Y
PJ07623			Y
PJ07630			Y
PJ07633			Y
PJ07647			Y
PJ07654			Y
PJ07668			Y
PJ07669			Y
PJ07671			Y
PJ07687			Y
PJ07697			Y
PJ07705			Y
PJ07705			Y
PJ07706			Y
PJ07707			Y
PJ07711			Y
PJ07733			Y
PJ07749			Y
PJ07751			Y
PJ07753			Y
PJ07762			Y
PJ07763			Y
PJ07776			Y
PJ07778			Y
PJ07779			Y
PJ07810			Y
PJ07868			Y
PJ07875			Y
PJ07875			Y
PJ07914			Y
PJ07915			Y
PJ07926			Y
PJ07927			Y
PJ07932			Y
PJ07937			Y
PJ07939			Y
PJ07951			Y
PJ07953			Y

Table 54 (Page 19 of 19). APAR Fixes in OS/2 2.x			
APAR	OS/2 2.00.1	Service Pak XR06055	OS/2 2.1
PJ07956			Y
PJ07961			Y
PJ07968			Y
PJ07973			Y
PJ08000			Y
PJ08007			Y
PJ08009			Y
PJ08020			Y
PJ08024			Y
PJ08035			Y
PJ08069			Y
PJ08091			Y
PJ08099			Y
PJ08110			Y
PJ08124			Y
PJ08127			Y
PJ08129			Y
PJ08131			Y
PJ08141			Y
PJ08155			Y
PJ08155			Y
PJ08172			Y
PJ08182			Y
PJ08184			Y
PJ08187			Y
PJ08193			Y
PJ08196			Y
PJ08227			Y
PJ08229			Y
PJ08241			Y
PJ08261			Y
PJ08284			Y
PJ08309			Y
PJ08330			Y
PJ08334			Y
PJ08340			Y
PJ08454			Y
PJ08455			Y
PJ08467			Y
PJ08546			Y
PJ08581			Y

Appendix E. OS/2 2.0 Volumes 1-5: Clarifications and Corrections

In conjunction with the availability of OS/2 2.0 we published a set of four volumes of information on the new environment. We later added a fifth volume on the print subsystem. In any effort to produce up-to-date information in a timely manner some material may need clarification or correction from time to time. This appendix contains these clarifications and corrections for OS/2 2.0 Volumes 1-5.

E.1 OS/2 Version 2.0 - Volume 1: Control Program, GG24-3730

Page 26, section 2.4.2.3, actually describes the concept of "Disk Frames." There is no concept of "Swap Pages" in the system.

For an improved description of disk frames, see E.6, "Disk Frames - Expanded Description" on page 362.

Page 41, section 3.4.3, second paragraph, add to the end of the paragraph, "DosKillThread only kills threads in its own process."

Page 64, second paragraph, "All programs that were started ..." should read "All programs that were started **and registered in the task list** ..."

Page 102, second paragraph (item 4), "This partition should be marked startable." should read "This partition should be marked **bootable**."

E.2 OS/2 Version 2.0 - Volume 2: DOS and Windows Environment, GG24-3731

Page 82, the third bullet referring to VEGB.SYS should be ignored and removed.

Sections 8.9, 8.10 and 8.11 address the OS/2 2.0 print subsystem. An up-to-date description of these sections can be found in *OS/2 Version 2.0 - Volume 5: Print Subsystem*. Of particular note are:

- In Volume 2, Figure 42 on page 158 does not address when the WIN-OS/2 spooler is disabled. Figure 97 on page 130 of this publication is correct and should be used instead.
- In Volume 2, Figure 43 on page 162 does not show the path for PostScript fonts to a PostScript printer when the fonts are not installed as downloadable. Figure 42 on page 76 of *OS/2 Version 2.0 - Volume 5: Print Subsystem* is correct and should be used.
- In Volume 2, section 8.10.2 on page 161 does not take into account that some font files with the same extension are different. Section 4.4 on page 74 of *OS/2 Version 2.0 - Volume 5: Print Subsystem* is correct and should be used.

E.3 OS/2 Version 2.0 - Volume 3: Presentation Manager and Workplace Shell, GG24-3732

No clarifications or corrections to date.

E.4 OS/2 Version 2.0 - Volume 4: Application Development, GG24-3774

Section 7.2 on page 103 refers to "messages" that are communicated by the Workplace Shell. This section describes internal system messages that are not available to applications and should not be confused with the "messaging" function that is used to interact between the system and application programs.

This publication is entitled *OS/2 Version 2.0: Application Development* but should not be interpreted as a replacement for *Application Design Guide, S10G-6260* in the OS/2 2.0 Technical Library - Version 2. A more appropriate title for Volume 4 may be *OS/2 Version 2.0: Writing Applications*.

E.5 OS/2 Version 2.0 - Volume 5: Print Subsystem, GG24-3775

No clarifications or corrections to date.

E.6 Disk Frames - Expanded Description

A virtual memory system is one that can address a memory space larger than that available in the system's physical memory. This is achieved by having memory blocks of data not currently in use stored onto the disk. In a system that support paging, fixed size memory blocks called pages are used, and for the 80386 and 80486, each page is 4KB.

OS/2 Version 2 uses both virtual memory and paging. To make room for new pages required to be brought into the limited physical memory, pages that are Least Recently Used (LRU) can be swapped out to disk in a swap file called **SWAPPER.DAT**.

Note

The disk partition on which the SWAPPER.DAT resides and its initial size are specified by the SWAPPATH setting in the CONFIG.SYS.

The swap file is managed by the file system like any other ordinary files. The swap manager which is part of the OS/2 kernel, uses a **bitmap** to manage the space within the swap file. The bitmap is stored in the physical memory and is set up when the system starts up. Each bit in the bitmap corresponds to a **Disk Frame (DF)** in the swap file as shown in Figure 199 on page 363 for a 2MB swap file.

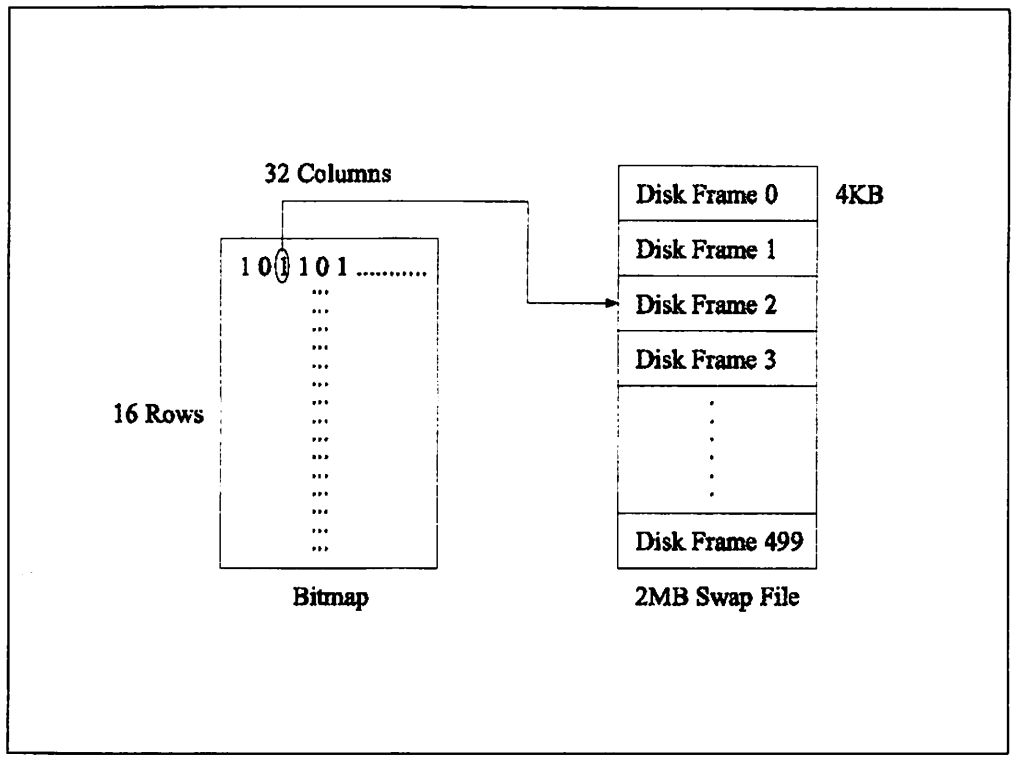


Figure 199. Disk Frame Management

If a bit is set ("1"), it means that the corresponding disk frame is occupied by the "swapped out" page, otherwise, the disk frame is empty and a page can be swapped out of physical memory and stored in it. The index in the bitmap of a disk frame is used as a swapID when a page is marked swappable. The swap file is an ordinary file allocated by the file system, and as such does not need to be contiguous on the disk.

List of Abbreviations

ABIOS	Advanced BIOS
ACL	Access Control List
APAR	Authorized Program Analysis Report
API	Application Programming Interface
APM	Advanced Power Management
ATM	Adobe Type Manager
BIOS	Basic Input/Output System
CD-DA	Compact Disk - Digital Audio
CD-ROM	Compact Disk - Read Only Memory
CD-ROMIXA	Compact Disk - Read Only Memory / Extended Architecture
CGA	Color Graphics Adapter
CUA	Common User Access
DBCS	Double Byte Character Set
DDE	Dynamic Data Exchange
DEM	DOS Emulation
DLL	Dynamic Link Library
DMA	Direct Memory Access
DPMI	DOS Protected Mode Interface
DMQS	Display Mode Query and Set
EGA	Enhanced Graphics Adapter
FAT	File Allocation Table
GA	General Availability
GUI	Graphical User Interface
HPFS	High Performance File System
IME	Input Method Editor
IPC	Inter Process Communication
ISO	International Standards Organization
ITSC	International Technical Support Center
LDT	Local Descriptor Table
LCD	Liquid Crystal Display
MAVDM	Multiple Application Virtual DOS Machine
MCI	Media Control Interface
MIDI	Musical Instrument Digital Interface
MMIO	Multimedia I/O Services
MMPMI2	Multimedia Presentation Manager/2
MRI	Machine Readable Instruction
MSCDEX	Microsoft CD Extensions
MVDM	Multiple Virtual DOS Machine
NLS	National Language Support
OLE	Object Linking and Embedding
PCMCIA	PC Memory Card International Association
PDD	Physical Device Driver
PEL	Picture Element
PMDD	Presentation Manager Device Driver
PSIVP	PS/ValuePoint
SAVDM	Single Application Virtual DOS Machine
SBCS	Single Byte Character Set
SCSI	Small Computer Systems Interface
SOM	System Object Model
SPI	Stream Programming Interface
SVGA	Super Video Graphics Array
VCDROM	Virtual CD-ROM Driver
VDD	Virtual Device Driver
VDM	Virtual DOS Machine
VGA	Video Graphics Array
VPIC	Virtual Programmable Interrupt Controller
VXD	Virtual Device Driver (in Windows 3.1 Enhanced Mode)
XGA	Extended Graphics Array

Index

Numerics

A

B

C

D

O

P

X

GG24-3948-00

OS/2 2.1 Technical Update

GG24-3948-00

PRINTED IN THE U.S.A.