OS/2 Workplace Shell Configuration Techniques

Document Number GG24-4201-00

August 1994

International Technical Support Organization
Boca Raton Center

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 (August 1994)

This edition applies to OS/2 Version 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 ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

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

Abstract

This document describes interfaces to the Workplace Shell of OS/2 Version 2.1. It provides a discussion and examples of using the CONFIG.SYS, .RC files, .INI files and programs to install, customize and distribute the OS/2 Workplace Shell in a stand-alone and distributed environment.

This document was written for IBM Technical Professionals. Some knowledge of OS/2 is assumed.

(290 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM Technical Professionals to understand and manipulate the OS/2 Workplace Shell through the use of the CONFIG.SYS, .RC files, .INI files and programs. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/2 Version 2.1. See the PUBLICATIONS section of the IBM Programming Announcement for OS/2 Version 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.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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:

CUA	IBM
Operating System/2	OS/2
Presentation Manager	WIN-OS/2
Workplace Shell	

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

Microsoft, Windows	Microsoft Corporation
PostScript	Adobe Systems, Inc.
1-2-3, Lotus, Freelance, Freelance Graphics, AmiPro	Lotus Development Corporation.
CorelDRAW!	Corel Systems Corp.
Aldus Pagemaker	Aldus Corportation
The Norton Commander	Symantec
DeskMan/2, DM/2, VUEMan/2, DM2Image	Development Technologies, Inc.
DeScribe	DeScribe, Inc.

Preface

This document describes interfaces to the Workplace Shell of OS/2 Version 2.1. It provides a discussion and examples of using the CONFIG.SYS, .RC files, .INI files and programs to install, customize and distribute the OS/2 Workplace Shell in a stand-alone and distributed environment.

This document was written for IBM Technical Professionals. Some knowledge of OS/2 is assumed.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction”
- Chapter 2, “Changing the Desktop via the CONFIG.SYS”

  This chapter describes the CONFIG.SYS, and some of the statements in the CONFIG.SYS that can be used to change your desktop. It might be helpful to have a brief description of some of the lines in the CONFIG.SYS that directly or indirectly affect the Workplace Shell
- Chapter 3, “Using the Resource Files to Change the Desktop”

  This chapter describes how the .RC files are constructed, how to compile these files correctly, and different ways of using them to customize existing desktops and create new desktops. We will be looking at how to create customized desktops and install them on multiple systems so that all the systems will have the same desktop.
- Chapter 4, “Using .INI files to Change the Desktop”

  This chapter describes how you can safely configure the Workplace Shell using the .INI files and also discusses some of the contents of the INI files.
- Chapter 5, “CID and The WorkPlace Shell”

  This chapter describes various ways to customize the desktops of work stations by doing a CID install across a LAN.
- Appendix A, “ITSO DESKTOPS Utilities”

  This appendix describes how to switch from one desktop to another on the same system and how to move a desktop from one system to another.

- Appendix B, “Associating Your Data Files”

  This appendix describes Associations, what they are, how to use them and what they will do for you when used with the Workplace Shell.

- Appendix C, “RC Syntax Diagrams”

  This appendix describes the syntax of the OS/2 Resource File. (RC File)

- Appendix D, “Deskman/2”

  This appendix describes what we discovered during our testing of Deskman/2. This is in no way a complete users guide for this product but simply a report of what we found with the testing that we did with the product.

- Appendix E, “Alternate Shells”

  This appendix describes the two alternate shells, TShell and MShell, that are included on the diskette that is shipped with this manual.

Related Publications

The publications listed in this section are considered particularly suitable for discussions related to the topics covered in this document.

- *OS/2 2.0 Technical Library*
- *Dvorak's Guide to OS/2 Version 2.1,* SR28-4642
- *Automated Installation for CID Enabled OS/2 V2.x,* GG24-3783
- *OS/2 Version 2.0 Volume 1: Control Program,* GG24-3730

International Technical Support Organization Publications

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

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

Acknowledgments

The advisor for this project was:

Jerry A. Stegenga II
International Technical Support Organization, Boca Raton Center

The authors of this document are:

Gert Ehing
IBM Germany

Peter Kelley
IBM Australia

Ken Kuhn
IBM Canada

Juan van der Breggen
ISM South Africa

Jerry A. Stegenga II
IBM United States

This publication is the result of a residency conducted by the International Technical Support Organization at the Boca Raton Center.

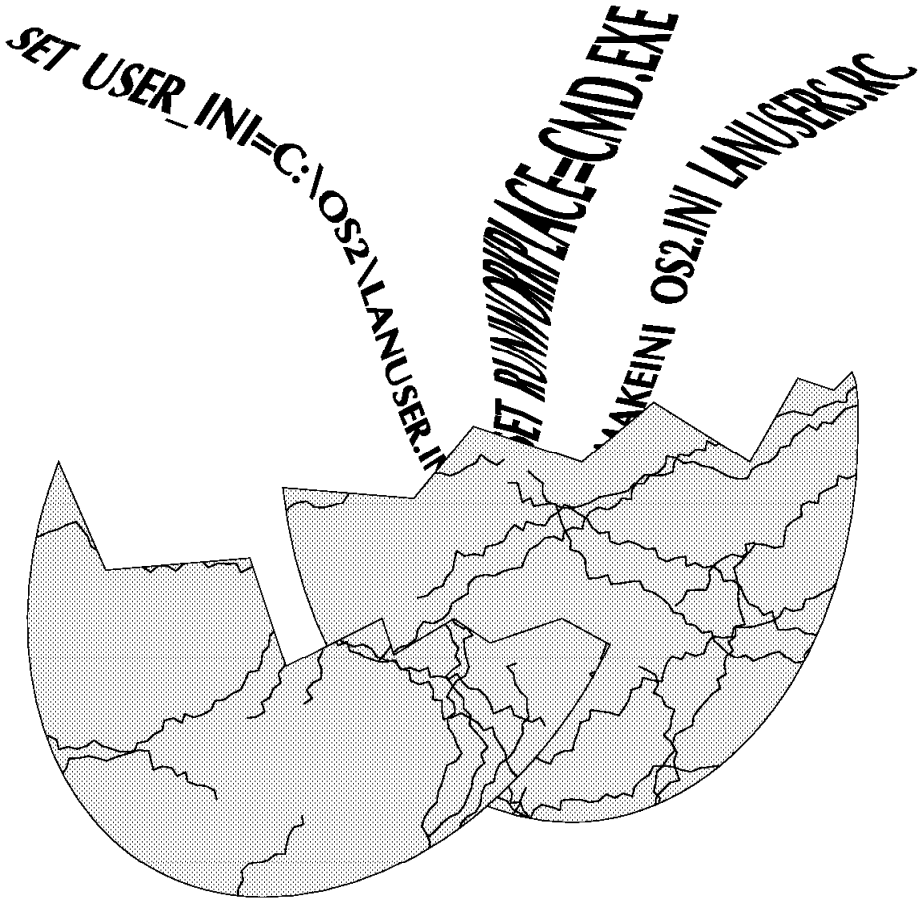
Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Tim Sennitt
International Technical Support Organization, Boca Raton Center

Monte Copeland
IBM Boca Raton Programming Center

Chapter 1. Introduction

The Operating System/2*(OS/2)* Workplace Shell* was designed with the intent to place all of the configuration and management of the system at the users fingertips. This is a blessing for the stand-alone power user but can become a nightmare for the coordinator of the distributed environment of today's corporations. This book will attempt to peel back some of the parts that make up the OS/2 Workplace Shell which will enable you to customize, distribute and maintain the "look and feel" that you want.



For example, in the case of a user workstation, you may want to remove the Shredder and the Desktop Pop-up Menu that now contains System Setup. Users may also want to access their files directly through a data file in a folder instead of the sometimes cryptic approach of opening the file from within a program. We have simulated this type of a desktop in Figure 1 on page 3.

In the case of a systems administrator workstation, you want total access to the entire OS/2 operating system and its setup and configuration functions. We have simulated this type of a desktop in Figure 2 on page 3.

For ease of maintenance there are two additional points to consider. The first is how to distribute or install a uniform customized desktop across all of your user workstations. The second is how systems administrators can access their more powerful desktop from any user workstation without having to reboot the machine.

We will approach all of this from the following perspectives:

- The system configuration file (CONFIG.SYS)
- The OS/2 resource files (*.RC)
- The OS/2 initialization files (OS2.INI and OS2SYS.INI)
- Programming interfaces (REXX and C)

We will explore each of these individually and then collectively.

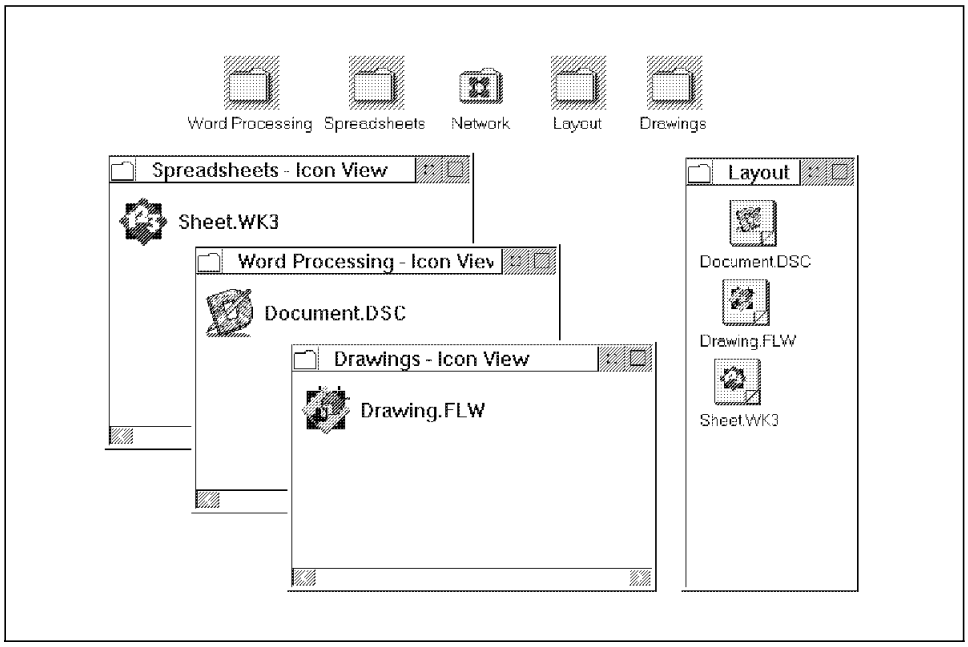


Figure 1. A User's Desktop

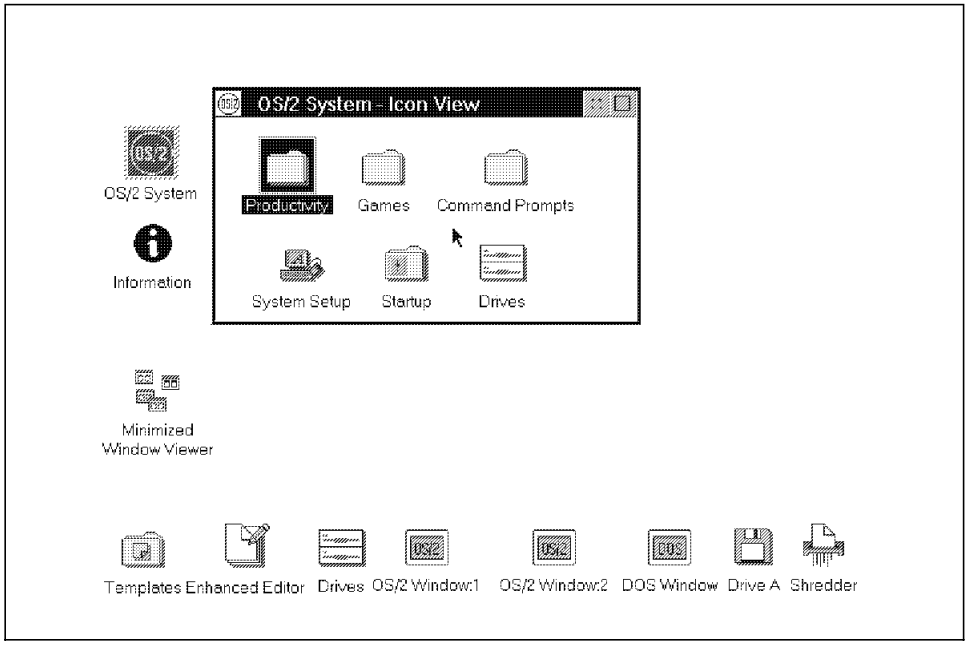


Figure 2. An Administrator's Desktop

Chapter 2. Changing the Desktop via the CONFIG.SYS

This section discusses the CONFIG.SYS, and some of the statements in the CONFIG.SYS that can be used to change your desktop. We will begin with a brief description of some of the lines in the CONFIG.SYS that directly or indirectly affect the Workplace Shell.

PROTSHELL= Loads the user interface program.

SET USER_INI= Indicates the name and location of the user INI file which contains the desktop setup information.

SET SYSTEM_INI= Indicates the name and location of the system INI file which will contain the information on what type of hardware is installed in the system and how it is configured.

SET OS2_SHELL= Starts the system loading the command line shell which is indicated in the SET COMSPEC= statement.

SET AUTOSTART= Contains a list of the components of the Workplace Shell that will be started when the system comes up. There are four possible options:

- PROGRAMS - Allows application programs, that were running when the system was shut down, to restart providing the SET RESTARTOBJECTS=NO statement is not used.
- TASKLIST - Enables the OS/2 Window list.
- FOLDERS - Opens the desktop folder which in turn will cause any other application that was running to restart unless the PROGRAMS option has been removed or the RESTARTOBJECTS= is specified and is set to NO or STARTUPFOLDERSONLY.
- CONNECTIONS - Restarts any network connections that were being used when the system was shutdown.

SET RUNWORKPLACE=Indicates which program to load as the Workplace and where it is located.

SET RESTARTOBJECTS=This statement is *not* included in the CONFIG.SYS during installation but it is defaulted to YES. This means the system will bring up any and all applications that were running when it was shut down. If you do not want to restart everything, this statement can be added to the CONFIG.SYS with one of the three possible parameters.

- NO - Do not restart any objects
- STARTUPFOLDERSONLY - Only restart objects in the Startup folder
- REBOOTONLY - Restart all objects only after a reboot or a power up, do not restart the objects if only the Workplace Shell is being reloaded

SET COMSPEC= Indicates which command line shell to load and where it is located.

LIBPATH= Lists the paths the system uses to locate DLL files.

SET PATH= Lists the directories where the system looks to locate programs and commands.

SET DPATH= Lists locations of data files of programs that are designed to use DPATH.

The rest of the CONFIG.SYS is made up of device driver statements, hardware and network definitions. These do not really have an effect on what you see as your desktop.

Tip

Before we started our testing we decided to add a DOS boot partition to our test system to eliminate the need and delay of booting the system from diskette. Of course in order for this to work we had to run with FAT partitions. This proved to be a wise decision and saved a lot of time waiting for the system to boot from diskette so we could recover from an experiment that went beyond the edge.

See 2.4.1, “OS/2 Full Screen Interface” on page 14 for a way to do this with HPFS partitions as well.

2.1 Desktop Loading Statements

There are two statements in the CONFIG.SYS which can be used to actually change what the system loads as your desktop or interface. They are:

- PROTSHELL=
- SET RUNWORKPLACE=

If you use either of these statements to load an alternate desktop you will get the following advantages and disadvantages:

- Advantages
  - User access to only desired application(s) and files, not system or setup functions
  - Possible performance increase due to lack of the PMSHELL overhead
- Disadvantages
  - Running applications will not restart automatically
  - No Shutdown or Lockup option due to no PMSHELL
  - Window List only displays with keyboard action (Ctrl Esc)

Since the advantages provide an environment that is easier to use by the average user and easier to maintain by the systems administrator, we looked at ways to get around the disadvantages.

2.1.1 Restarting Applications

We tested to see if applications that were running when you shut down your system would automatically start when the system was restarted with other than PMSHELL. We left open, and running, a network, CM/2 and several applications. We then made various changes to the desktop loading statements in the CONFIG.SYS and rebooted. We found that none of the previously running applications would restart. We checked the SET AUTOSTART= statement in the CONFIG.SYS to insure it was set to the default ″PROGRAMS,TASKLIST,FOLDERS,CONNECTIONS″ and we did not add the SET RESTARTOBJECTS= statement.

To work around this it is necessary to use a STARTUP.CMD file to start the network, CM/2 or any other application you wish to have initially running if you use anything other than a fully functional shell such as PMSHELL or MSHELL (see Appendix E, “Alternate Shells” on page 271) as your desktop.

2.1.2 Shutdown Alternatives

Shutdown is a very important aspect of OS/2. Shutdown is something that should be done so that you will have an orderly closing of your system rather then simply powering the system off. This is especially important if you are using HPFS. We modified the CONFIG.SYS to load a PM application as an alternative to PMSHELL. We then tested several of the stand-alone shutdown programs that are available and found that, because they are not part of the alternative shell, they closed the alternative shell just like it was another program. This meant that the alternative shell would then attempt to restart before shutdown was complete. This process led to intermittent, unpredictable and sometimes perpetual results.

What we needed was for the alternative shell to be the last thing that was closed by the system. The answer was quite simple. If you want a shell to be the last thing that is shut down, then make the shutdown program your shell. However, if you do that with the current shutdown programs, as soon as your system comes up, it will attempt to shut down. So what we needed was a special shutdown program that lay dormant until you wanted to execute it. We have provided such a program on the diskette included with this book. This program provides an orderly shutdown similar to the desktop shutdown process. There is an additional reason for making the shutdown program the shell; whatever you define as the shell cannot be closed. If you try to close it, it will just restart. Using shutdown as your shell assures you that a user will always be able to perform a shutdown.

2.1.3 PROTSHELL=

The PROTSHELL statement is executed first and it points to the user interface program and command processor that will be loaded when the system starts up. The default program that is called from the PROTSHELL statement after installing OS/2 2.1 is PMSHELL.EXE, which loads the default Presentation Manager* Shell and its desktop.

It is *possible* to call other programs to be the shell and then possibly load additional programs depending on what you have chosen for your shell.

Recommendation

Based on our testing, we recommend that you *do not* change or delete the PROTSHELL= statement. There is one exception and it is described in 2.4.1, “OS/2 Full Screen Interface” on page 14.

Notes

It is not possible to start Windows** or DOS programs as your desktop by using the PROTSHELL statement.

If you use a PM program for your desktop, the current background desktop color will still be active. However, if you have a bitmap as the background for your desktop it will not be displayed.

2.1.4 SET RUNWORKPLACE=

The SET RUNWORKPLACE= statement can be used to load a program, other than the default PMSHELL.EXE, as your desktop and is a relatively stable way of doing this. The RUNWORKPLACE statement will allow you to load some, but *not all*, DOS, Windows or OS/2 applications as your desktop. An interesting side effect to doing this is that if the application that is loaded in this statement is closed it will automatically reopen because it *is* the Workplace Shell. As mentioned in the introduction to this section, you will also have no Shutdown option as the desktop menu will not be available. These two combined could give you a *"perpetual"* desktop, that is, a desktop you can never close. We will show you how to work around this problem in the following examples.

Note

Some system settings, such as defined printers and installed fonts, will be available to applications started from the RUNWORKPLACE statement. Other system settings, such as the background bitmap, will not.

2.2 Single Application Access

The most basic desktop you could run would be loading a single application as your desktop. If you have users who only run one application on their system all the time then you might want to make use of this setup. It will provide better performance as you will not have the overhead of the Workplace Shell. If the application you run from does not allow you to start any additional programs you will only be able to run the single application on your system. If the application you choose does, however, have multiple threads for tasks such as printing then you will be able to take advantage of them.

In our testing, we first tried to simply change the SET RUNWORKPLACE= statement to point to the desired application. While this did work in most cases, it gave us the perpetual desktop we described earlier. We found the only way to create a stable application desktop as an alternative to PMSHELL was to start any and all applications you desire from a STARTUP.CMD file and use the PROTSHELL= statement to load our shutdown program. We also remarked out the SET RUNWORKPLACE= statement. The two lines in the CONFIG.SYS we used for the following examples are shown in Figure 3 The structure of the STARTUP.CMD file that we used is shown in Figure 4.

```
PROTSHELL=SHUTDOWN.EXE
Rem SET RUNWORKPLACE=PMSHELL.EXE
```

Figure 3. CONFIG.SYS Modification for Single Application Environment

```
START D:\directory\program.EXE
EXIT
```

Figure 4. Structure of STARTUP.CMD for Single Application Environment

Notes

While the use of START is not necessary in the single application environment it is a good habit to get into now for the other scenarios.

If you do not use EXIT at the end of the STARTUP.CMD you will have an OS/2 Window named STARTUP.CMD, along with the application, on your desktop.

If you close the application that was started in STARTUP.CMD then you will not be able to reopen it as there is no program object on the desktop.

As we mentioned at the start of this section, *most* applications worked transparently and did not require any extra steps. We have found some exceptions in our testing, however, and there are surely more.

2.2.1 Using AmiPro for OS/2 as a Desktop

Other than not having a shutdown option, using the SET RUNWORKPLACE statement to load AmiPro** for OS/2 had no problems. But since we wanted a system we could shut down cleanly we modified our CONFIG.SYS as shown in Figure 3 on page 10 and created a STARTUP.CMD file to launch AmiPro for OS/2 as shown in Figure 6.

```
START D:\AMIPRO\AMIPRO.EXE
EXIT
```

Figure 5. STARTUP.CMD to Load AmiPro for OS/2

2.2.2 Using DeScribe as a Desktop

In addition to not having a shutdown option, setting the RUNWORKPLACE statement to load DeScribe** had one minor problem. When DeScribe loads from the RUNWORKPLACE it looks for a file called *"-"* in the root directory of the boot drive. If it does not find the file the system will display a message indicating it can not find the file and do you wish to create the file, Yes or No. If you create the file the system will no longer have this problem in the future. However, if you choose not to create the file you will get the error message each time you reload the Workplace Shell which is now DeScribe.

So, instead, we modified our CONFIG.SYS as shown in Figure 3 on page 10 and created a STARTUP.CMD file to launch DeScribe as shown in Figure 6.

```
START D:\DESCRIBE\DESCRIBE.EXE
EXIT
```

Figure 6. STARTUP.CMD to Load DeScribe

2.2.3 Using CorelDRAW! as a Desktop

Some programs require more than just running their executable in order to start properly. One example of this is CorelDRAW!** for OS/2. CorelDRAW! would not start directly from the RUNWORKPLACE statement. If you try this you will receive two or three errors in succession. These errors are:

- Config file not found. Using built in default (Occurs sometimes)
- Error opening \CORELDRW.PAL File. (Occurs all the time)
- Error opening \CORELDRW.INK File. (Occurs all the time)

After selecting the OK button in the last panel, the Workplace Shell, which in this case is CorelDRAW!, would close and then start up again with the same results.

All of this is because CorelDRAW! *requires* that you be in the COREL32 directory in order to start the program. A simple entry in the PATH statement will not work. In the normal desktop environment this is accomplished by setting the Working directory in the Program object to the proper path. Since we are not using PMSHELL the system does not have access to those settings. We must therefore set the environment through commands.

Figure 7 shows a simple CMD file we used to load CorelDRAW!. These statements:

- Change to the CorelDRAW! drive
- Change to the CorelDRAW! directory
- Start CorelDRAW!

Naming this .CMD file something other than STARTUP.CMD and calling it from the SET RUNWORKPLACE statement in the CONFIG.SYS would cause an OS/2 window, with the title Workplace Shell, to open. The CMD file would actually execute within this window and then the window remains displayed but is not available to the user.

Again, the best solution was to use a STARTUP.CMD file to load CorelDRAW!. However, in this case, if we simply put START D:\COREL32\CORELDRW.EXE in a STARTUP.CMD file we would get the same errors decribed previously. As we mentioned, CorelDRAW! needs to be launched from the directory where it is installed. You can use the example in Figure 7 to do this.

```
D:
CD COREL32
START CORELDRW.EXE
EXIT
```

Figure 7. STARTUP.CMD to Load CorelDRAW!

2.2.4 Using Aldus PageMaker for Windows as a Desktop

We encountered no problems when we modified our CONFIG.SYS as shown in Figure 3 on page 10 and created a STARTUP.CMD file to launch Aldus PageMaker** for Windows as shown in Figure 8.

```
START D:\PM4\PM4.EXE
EXIT
```

Figure 8. STARTUP.CMD to Load Aldus Pagemaker for Windows

2.3 Multiple Application Access via File Managers

If you load an application such as the WIN-OS/2* File Manager or Norton Commander** for OS/2 then you will be able to access multiple applications from within them. Other ways to load multiple applications on your OS/2 system, without loading the default Workplace Shell, include using a STARTUP.CMD file or an OS/2 command prompt. These will be discussed in following sections.

2.3.1 Using the WIN-OS/2 File Manager as a Desktop

When we tried loading the WIN-OS/2 File Manager as our Workplace Shell using SET RUNWORKPLACE in the CONFIG.SYS it would come up with an Application Execution Error. However, when we selected the OK button in the error dialog the Windows File Manager was ready to use. So, instead, we modified our CONFIG.SYS as shown in Figure 3 on page 10 and created a STARTUP.CMD file to launch the WIN-OS/2 File Manager as shown in Figure 9. Remember that you can start most OS/2 PM, Windows *and* DOS applications directly from the WIN-OS/2 File Manager.

```
START C:\OS2\MDOS\WINOS2\WINFILE.EXE
EXIT
```

Figure 9. STARTUP.CMD to Load WIN-OS/2 File Manager

2.3.2 Using Norton Commander as a Desktop

Starting Norton Commander from the SET RUNWORKPLACE statement in the CONFIG.SYS worked fine and the only problem we noticed was the same shutdown problem that was occurring on all of the Workplace Shell tests. So, instead, we modified our CONFIG.SYS as shown in Figure 3 on page 10 and created a STARTUP.CMD file to launch the WIN-OS/2 File Manager as shown in Figure 9 on page 13. Remember that you can start most OS/2 PM, Windows *and* DOS applications directly from the Norton Commander File Manager.

```
START D:\NCPM\NCPM.EXE
EXIT
```

Figure 10. STARTUP.CMD to Load Norton Commander's File Manager

2.4 Application Access via CMD.EXE

You can use the PROTSHELL and RUNWORKPLACE statements to create a command line driven environment, either full screen or windowed. The full screen environment gives you the best possible performance but restricts you to one application. You also cannot access PM, DOS or WIN-OS/2 applications. The windowed interface, however, lets you navigate through almost the entire OS/2, PM, DOS and WIN-OS/2 system without the overhead of the PMSHELL.

2.4.1 OS/2 Full Screen Interface

To get an OS/2 Full Screen prompt as your interface modify the PROTSHELL and SET RUNWORKPLACE lines in your CONFIG.SYS as shown in Figure 11.

```
PROTSHELL=C:\OS2\CMD.EXE
rem SET RUNWORKPLACE=C:\OS2\PMSHELL.EXE
```

Figure 11. CONFIG.SYS to Load an OS/2 Full Screen as the Desktop

When we loaded CMD.EXE in the PROTSHELL the system came up with an OS/2 Full Screen. If you have a STARTUP.CMD file that starts applications it will execute any OS/2 non-PM programs but PM programs will generate SYS1059 errors "The system could not execute the specified program."

The print functions continued to operate normally regardless of whether the WORKPLACE statement was included in the CONFIG.SYS or not. Note that ″normally″, in the case of a command line, means that you must specify a port.

2.4.2 OS/2 Window Interface

To get an OS/2 Window as your interface modify the PROTSHELL and SET RUNWORKPLACE lines in your CONFIG.SYS as shown in Figure 12.

```
PROTSHELL=C:\OS2\PMSHELL.EXE
SET RUNWORKPLACE=C:\OS2\CMD.EXE
```

Figure 12. CONFIG.SYS to Load an OS/2 Window as the Desktop

Using CMD.EXE as the RUNWORKPLACE caused an OS/2 Window to be displayed which was titled Workplace Shell. From this window we could issue START commands and launch multiple applications. We could even execute START PMSHELL.EXE and get the OS/2 Workplace Shell to come up. The printing functions worked provided they were setup prior to loading the new WORKPLACE. As mentioned before, if we closed this window it simply reopened again. To shut the system down we simply ran our shutdown program from the window.

2.5 Multiple Application Access via .CMD Files

In 2.2, “Single Application Access” on page 9 we showed you how to start single applications from the SET RUNWORKPLACE and the STARTUP.CMD file. A more typical environment is to have multiple applications and, possibly, communications programs running at the same time. This can be accomplished from the file managers and the windowed command prompt we mentioned before. However, if you are not using PMSHELL and you want them to automatically start with the system then you must use a .CMD file. You can either use the STARTUP.CMD file or you can call a user.CMD file from the SET RUNWORKPLACE statement in the CONFIG.SYS.

2.5.1 Multiple Application Access Using STARTUP.CMD

The STARTUP.CMD file has been around for a long time. Its use in this environment is basically the same as if you are using it with PMSHELL. There are, however, some things you will have to look out for.

Once again, we modified our CONFIG.SYS as shown in Figure 3 on page 10 to give us a way to cleanly shut down the system. We then created a STARTUP.CMD file to launch multiple applications as shown in Figure 13.

```
START D:\DESCRIBE\DESCRIBE.EXE
START D:\NCPD\NCPM.EXE
D:
CD COREL32
START CORELDRW.EXE
START F:\PM4\PM4.EXE
EXIT
```

Figure 13. STARTUP.CMD to Load Multiple Applications

If you want to start applications such as CM/2 or IBM* LAN Requester then you can include them in the STARTUP.CMD file as well. Figure 14 shows how they should look if they were added to our previous example. Notice that we did not use START before the NET START REQ command. If we did it would work but it would also leave an OS/2 Window on the desktop which is not what we desired.

```
START D:\DESCRIBE\DESCRIBE.EXE
START D:\NCPD\NCPM.EXE
D:
CD COREL32
START CORELDRW.EXE
START F:\PM4\PM4.EXE
START C:\CMLIB\CMSTART.EXE
NET START REQ
EXIT
```

Figure 14. STARTUP.CMD to Load Multiple Applications and Communications

2.5.2 SET RUNWORKPLACE=user.CMD

You can use the SET RUNWORKPLACE to run a .CMD file as your desktop but heed the following warning!

Warning!

Do not use the START command on the last line in the CMD file to load an application. If you do, the START command will close the window after it has executed. Due to the fact the window is the Workplace Shell it will reopen and execute the CMD file again and continue to loop, trying to start the application over and over again! The way to get around this is to have the last line in the CMD file execute something other than a START command.

Depending on which application we actually loaded from the CMD, we saw different results on the display.

- If we were loading CorelDRAW! we found that the loop would result in numerous CorelDRAW! sessions being opened. We found that if we let the system loop on its own we eventually had 45 CorelDRAW! sessions up on the display and the Workplace Shell window was still executing trying to start more but the system would not open any more.
- If we were loading DeScribe we found that the loop would result in the Workplace Shell window opening and closing and, after doing this for several minutes, a DeScribe logo screen would be displayed several times. It looked like DeScribe was having trouble starting due to the length of time it takes for DeScribe to come up and the system was busy opening additional OS/2 windows and trying to launch DeScribe numerous times.

Having the Workplace Shell starting multiple applications did cause a problem with standard shutdown programs. Before the shutdown could complete the applications were all starting again. The shutdown tried to continue closing these applications but eventually it would fail and leave multiple occurrences of the applications on the desktop. Again the safest method is use the STARTUP.CMD along with making the shutdown program your shell.

2.6 WIN-OS/2 as Your Desktop

In order to use WIN-OS/2 as your interface to the system you must use a .CMD file. Again we recommend that you modify your CONFIG.SYS as shown in Figure 3 on page 10 to give you a way to cleanly shut down the system. Then create a STARTUP.CMD file to launch the WIN-OS/2 desktop as shown in Figure 15. This method will, in all cases, start a Full Screen WIN-OS/2 session. Any Windows applications that you start will be contained in this session because the seamless Windows drivers are not available when PMSHELL is not the active desktop.

```
START C:\OS2\MDOS\WINOS2\WIN.COM
EXIT
```

Figure 15. STARTUP.CMD to Load WIN-OS/2 as the Desktop

Starting a Windows session as the Workplace Shell by using SET RUNWORKPLACE did not function properly. The STARTUP.CMD would execute and all of the various applications we started from it would come up fine. There would be a DOS icon on the screen called Workplace Shell and if we selected this ICON we would go to a WIN-OS/2 Full Screen but there would be an error panel displayed which was an Application Execution Error and the following message was in it along with an OK button:

Cannot find file; check to ensure path and file name are correct

If we selected the OK button the existing session stops and then it will start again and the same error occurs. If we had some applications set to start automatically when the WIN-OS/2 session started, they would not even begin to start.

If we pressed Ctrl Esc the screen would switch to the Window List with the OS/2 applications on the Desktop and we were able to use any of the OS/2 applications that had been started. Also if there was an OS/2 command prompt running we could execute a START WINOS2.COM and a Windows Full Screen session would start. Any Windows applications set to automatically start would then start up fine in that session.

Chapter 3. Using the Resource Files to Change the Desktop

Many of you are aware of the .INI files and that they contain information about your system and the desktop. What you may not be aware of is how they are created and how you can create your own. We are sure that there are some people who would like to create or modify a desktop and then install that desktop on a number of machines. After installation of a program on the LAN you might want to update all the systems on your LAN so that all the desktops have the same objects. This can be accomplished using the OS/2 Resource Files (.RC).

The .RC files are the source code from which the system creates the .INI files. We can use .RC files to create new .INI files from which a new Workplace Shell can be built. We can also use the .RC files to change our existing Workplace Shell. We can create, update or delete objects that appear on our desktop, change the way objects behave and change color settings.

As most PM programmers know, we use resource files when we code a PM program to define the menus, icons, strings and other settings. The .RC files that we are discussing in this chapter are not like the resource files used in PM programming. They are not programmed or used in the same way as in PM programming. The .INI .RC files use commands very similar to commands used in REXX. If you have programmed the desktop using REXX you will be familiar with the commands and keywords in the .RC files.

We will be looking at how the .RC files are constructed, how to compile these files correctly, and different ways of using them to customize existing desktops and create new desktops. We will be looking at how to create customized desktops and install them on multiple systems so that all the systems will have the same desktop.

3.1 The Default .INI Files

The OS/2 system uses primarily two .INI files. These .INI files contain information for your Workplace Shell. Some programs use their own .INI files such as the Enhanced Editor, which uses EPM.INI. OS/2 has a user .INI file (OS2.INI) and a system .INI file (OS2SYS.INI). We will refer to them as OS2.INI and OS2SYS.INI for the remainder of this section. If you look at your CONFIG.SYS you would find two lines which set these .INI files. For example, if you installed OS/2 on your C: drive then the lines will look as follows:

```
SET USER_INI=C:OS2OS2.INI
SET SYSTEM_INI=C:\OS2\OS2SYS.INI
```

3.2 The Default .RC Files

After you have installed your OS/2 system you should have seven .RC files in the OS2 directory:

- INI.RC
- INISYS.RC
- OS2_13.RC
- OS2_20.RC
- WIN_30.RC
- UPINI.RC
- LOCK.RC

The files INI.RC and INISYS.RC are the two files that compile into OS2.INI and OS2SYS.INI respectively. The other files do not have complete structures to be compiled as new .INI files. They are meant to be compiled on top of an existing *user* .INI file, such as the OS2.INI.

Warning!

There is only one .RC file to create the OS2SYS.INI and that is the INISYS.RC file. None of the other .RC files can be compiled over the OS2SYS.INI. All the other files were designed to be compiled over OS2.INI. We do not recommend that you write .RC files to be compiled over OS2SYS.INI as we have found that it puts the system in a very unstable state.

3.3 The MAKEINI Compiler

To compile your .RC files into .INI files you have to use the MAKEINI.EXE compiler.

To use the MAKEINI compiler you need a valid .RC file for the source file. Remember that the target .INI file cannot be an .INI file that is currently being used by the system. To compile a .RC file:

1. Open an OS/2 Window or an OS/2 Full screen prompt.
2. Change to the directory where the .RC and .INI files are. These files are usually located in the OS2 directory.
3. Type **MAKEINI target.INI source.RC**

The MAKEINI compiler is a very basic compiler and it has limited feedback regarding errors. We have found that only in some instances will it report a syntax error, but without any line numbers of where the error is. The error message that you will see is

```
MAKEINI.EXE - Error in INI file
```

This could mean two things:

- There is a syntax error in your .RC file.
- There is a problem in creating the new .INI file.

We have found that the most common mistake is to compile an .RC file to an .INI file that is currently being used by OS/2.

Notes

OS/2 locks the current .INI files that are being using for the desktop. This prevents the user from changing or deleting them. You have to boot from the OS/2 Install Diskette and Diskette 1 and *then* do a MAKEINI if you want to change the .INI file that is specified in the CONFIG.SYS.

You can use MAKEINI to create a .INI with a different name and change the `SET USER_INI=` statement in the CONFIG.SYS, or use the PrfReset API call to use the new .INI file. See Appendix A, “ITSO DESKTOPS Utilities” on page 217 for more information on using PrfReset to switch desktops.

With OS/2 Version 2.1 you are able to *copy* the .INI files that are being currently used by the system. Remember, however, that the .INI files may not be up to date while the system is running. Only after a shutdown can you be assured that they contain all of the current information.

The other error message that you might see is when you try to use your new .INI file. You may get a message box saying that the .INI file is corrupted and the shell has stopped. This error message will occur more often than the error you can get while using the MAKEINI compiler. You cannot continue from this error message. You *will* get an OS/2 Window when you click on the OK or Cancel buttons but there is no way that you can do a shutdown or get back to your old desktop.

Note

It is advisable not to delete the original desktop directory DESKTOP. Also, make a backup of the original OS2.INI and OS2SYS.INI files and use them when you want to use the original desktop. If you forget to do this remember that there is a set of the original, after installation, .INI files in the \OS2\INSTALL directory.

3.3.1 Making a New .INI File

If you are going to create a *new* user .INI or a *new* system .INI file, it is necessary to either delete the old .INI file by the same name or use a new name when creating the new .INI file. We advise that you give the new .INI file a unique name to avoid any problems when you want to use the new .INI file. If you don't delete the old file you will compile the new .RC file over the old .INI file, and your new desktop may not be what you want. What we found was that some of the objects changed when we compiled the new .RC file without deleting the old INI file. This caused some confusion as we made changes to icons in the .RC file and it was not changed on the new desktop.

The other problem we found is when we do delete an existing .INI file and create a new .INI file with the same name the system gave us an error after we rebooted. See Figure 16 on page 23. This error occurred randomly and it happens because the new user .INI file tries to use the old desktop. We have found two ways of overcoming this problem:

- Delete the old desktop when we are using the same name for a new .INI file.
- Compile the new .INI file with a unique name. This will create a new desktop and a new desktop directory.

Remember that the desktop is a glorified folder. It has its own directory structure on the hard disk.

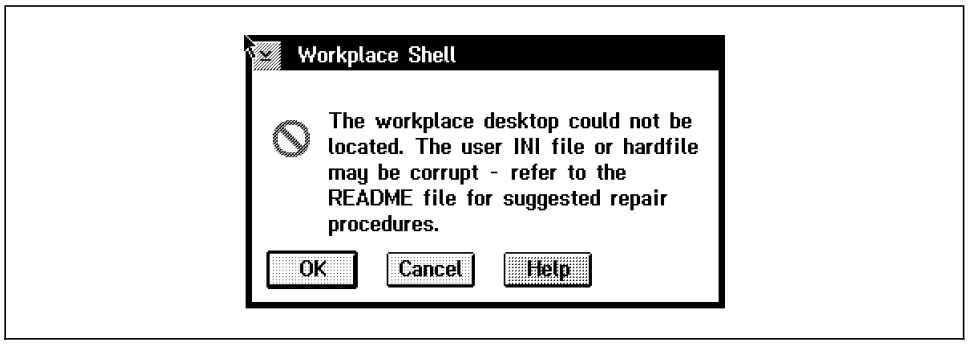


Figure 16. Workplace Shell INI Error

To compile an .RC file to a new .INI file:

1. Open an OS/2 Window or an OS/2 Full screen prompt
2. Change to the directory where the .RC and .INI files are. These files are usually located in the OS2 directory
3. Type **MAKEINI target.INI source.RC**

3.3.2 Changing an Existing .INI File

Instead of creating a new .INI file you may only want to change or append to an existing .INI file. We would then not delete the .INI file but compile the .RC file to the current .INI file. An example of this is when you want to change your desktop to look like a Windows or OS/2 1.3 desktop. You use the existing user .INI file and compile the Windows or OS/2 1.3 .RC files over the existing .INI file. To change your existing desktop to look like Windows:

1. Boot from the OS/2 Install disk and Disk 1
2. Press **ESC** at the Welcome screen
3. Change the drive to the boot hard drive

   This is usually drive C so you would type **C:**
4. Press **Enter**
5. Change the directory to OS2 by typing **CD OS2**
6. Press **Enter**
7. Type **MAKEINI OS2.INI WIN_30.RC**
8. Press **Enter**
9. Remove any diskette from the diskette drive
10. Reboot the machine by pressing **Ctrl-Alt-Del**

When your machine has finished booting the desktop will look like Windows 3.0. To change back to the OS/2 2.1 desktop you need to recompile the OS2.INI file from the INI.RC file with MAKEINI. If your target user .INI file does not exist you need to compile the default .RC file to create a user .INI file, and then compile the cutomized .RC file over that user .INI file. For example, if you wanted to create a new desktop that looks like OS/2 1.3 but you also wanted to keep the current OS/2 2.1 desktop, you would issue the following commands:

```
MAKEINI OS2_13.INI INI.RC
MAKEINI OS2_13.INI OS2_13.RC
```

Then, to use this new desktop that looks like OS/2 1.3, you must change the SET USER_INI=filename in the CONFIG.SYS and then reboot from this .INI file as follows:

```
SET USER_INI=C:OS2OS2_13.INI
```

Remember you always have to shutdown and reboot the system in order for changes to the CONFIG.SYS to take effect, and the new desktop to be created.

3.3.3 The LOCK.RC File

The LOCK.RC file is used to reset the password of your system. You would use this when you have Lock on Startup checked on the Lockup page of the desktop settings notebook and you have forgotten your password. The following steps will reset the password:

1. Boot from the OS/2 Install disk and Disk 1
2. Press **ESC** at the Welcome screen
3. Change the drive to the boot hard drive

   This is usually drive C so you would type **C:**
4. Press **Enter**
5. Change the directory to OS2 by typing **CD OS2**
6. Press **Enter**
7. Type **MAKEINI OS2.INI LOCK.RC**
8. Press **Enter**
9. Remove any diskette from the diskette drive
10. Reboot the machine by pressing **Ctrl-Alt-Del**

Note

This procedure clears out *all* of the Lock up options which includes the background bitmap and Lock on Startup as well as the password.

3.4 The Structure of .RC File Commands

All of the commands in the .RC files use three parameters. These parameters are strings enclosed in double quotes with a space between each string. For example:

```
"PM_InstallObject" "System Clock;WPClock;<WP_CONFIG>" "OBJECTID=<WP_CLOCK>"
```

The three parameters are defined as:

```
"Application" "Key" "Value"
```

Parameter Description

Application This parameter is used to tell the system where you want to make a change. This is usually a name of a group or application in the OS/2 system that contains objects. Examples of these applications are PM_Fonts, PM_System_Colors and PM_SPOOLER. One statement that is used differently to the others is the PM_InstallObject and we will discuss it in more detail later in this chapter. Table 1 on page 27 contains a list of the groups.

Key This parameter has the profile of the new object that is to be created or the object that needs to be changed. These are usualy a name of a object or a key to a setting of the Workplace Shell. These objects must be related to the group that you specify in the first parameter.

Value A string of settings or values for the objects refered to by the Key parameter. This could be, for example, a filename or a color setting.

You can use more than one setting in the Key and Value fields as long as they are separated with semicolons. The Application parameter always has only one statement.

Table 1. Application Parameters

Application Name	Description
PM_ASSOC_TYPE	Association type.
PM_DEVICE_DRIVERS	Printer device drivers.
PM_Font_Drivers	Presentation Manager Font Driver.
PM_Fonts	PM fonts with path and filename.
PM_INFO	Information about the OS/2 system such as the version level.
PM_IMAGECNV	The image conversion path.
SYS_DLLS	DLLs used by the system for example REXXINIT.
PM_SPOOLER	Printer spooler information.
PM_National	All the PM National settings.
PM_IBMBGA	PM 8514 adapter settings.
WINOS2	WIN-OS2 settings.
PM_Workplace	Special global settings for the Workplace Shell.
PM_Workplace:InstallGroups	Create group icons on the desktop.
PM_Workplace:InstallAutorun	Install programs that should run the first time the system comes up.
PM_Workplace:InstallDiskOnDesktop	Install the disk folder shadow on the desktop the same as the Drive A shadow that is default.
PM_ControlPanel	Global settings for the Workplace Shell. These are the settings in the System settings notebook.
PM_InstallObject	Creates new objects on the desktop.
PM_Colors	Settings for each object's color.
PM_Default_Colors	Settings for each object's default colors when you hit the Default button.
PM_DISPLAYDRIVERS	Display drivers for the display adapter.
PM_XXXXXX_Colors	Colors for each Scheme Palette. This information is in the INISYS.RC file.

Comments in the .RC files are the same as in C and REXX files. You start a comment with /* and end it with */ (See Figure 17). Comments can start before or after a command line or on a line of their own. Although comments can be placed inside a command line, and will cause no error during compiling, we do not advise it because it can be confusing. It may also cause an error when the new .INI file is used.

```
/* This is a comment */
```

Figure 17. A Comment Line in an .RC File

In the .RC files the boot disk is referred to as a question mark (?). When OS/2 boots this is replaced with the drive letter on which OS/2 is installed. This question mark is used because OS/2 can be installed on any drive, and the operating system must know where to find the files. When referring to a file you can use the drive letter, or the question mark (?) when the file is on the same drive that OS/2 is installed on (See Figure 18).

```
"PM_InstallObject" "Clipboard Viewer;WPProgram;<WP_TOOLS>"
   "EXENAME=?:\OS2\CLIPOS2.EXE;OBJECTID=<WP_CLIPV>"
```

Figure 18. Using ? as a Drive Letter

3.5 The User .RC File Structure

The INI.RC file can be divided into four sections.

- RC File Header
- System and International information
- Objects definitions
- Colors and display modes

These are all the sections of the INI.RC file. These sections or even part of these sections can be used to update an existing .INI file by using them in smaller, subset .RC files.

All sections must start with a STRINGTABLE and be enclosed in a BEGIN and END. The statement STRINGTABLE informs the compiler that we will be using a set of strings which contain the commands that we want to execute. If you use STRINGTABLE REPLACEMODE then the strings within the BEGIN,

END will *replace* existing objects. This option cannot be turned off inside the BEGIN, END statements. When you use STRINGTABLE without REPLACEMODE the objects will be created and changed, but not replaced, unless specified with a REPLACE option inside of the string.

3.5.1 RC File Header

The only section that must be in any .RC file is the .RC File Header. You can not compile a .RC file without this header. These headers are all the same.

The header must have a STRINGTABLE and a BEGIN followed by ten lines of two double quotes. The ten lines with the two double quotes are very important for the MAKEINI compiler. If you do not include these lines the compiler will go into an endless loop and you will end up with a corrupted .INI file. After the ten lines of quotes you can add your own lines of code.

```
CODEPAGE 850

STRINGTABLE
BEGIN

  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""

END
```

Figure 19. RC File Header

3.5.2 System Information

This area includes:

- The Presentation Manager(PM) setting
  - ControlPanel
  - DefaultSetup
  - Workplace
- Association types
- Workplace Shell settings
  - Printer drivers
  - Fonts
- System Settings

3.5.2.1 Presentation Manager Setting

The first line in the System Information area is the Border Width setting. This is done using the PM_ControlPanel Application value. The other PM_ControlPanel values are described in Table 3 on page 33. The width of the border can be set through the desktop by using the Scheme Palette in the System Setup folder. This interface allows you to set both a horizontal and vertical value from 1 to 50. These numbers represent the number of pixels (dots) on your screen that are used for the border. This is a system wide setting and cannot be changed for individual windows. The initial default is 4. (See Figure 20.)

In the .RC file you are limited to just one value which will be used for both the horizontal and vertical width; however, you can specify widths above 50. We tested this with a width of 150 but, although it worked, we failed to find a practical reason as to why you would want a border this wide.

```
"PM_ControlPanel"    "BorderWidth"      "4",
```

Figure 20. Default .RC File Border Width Setting

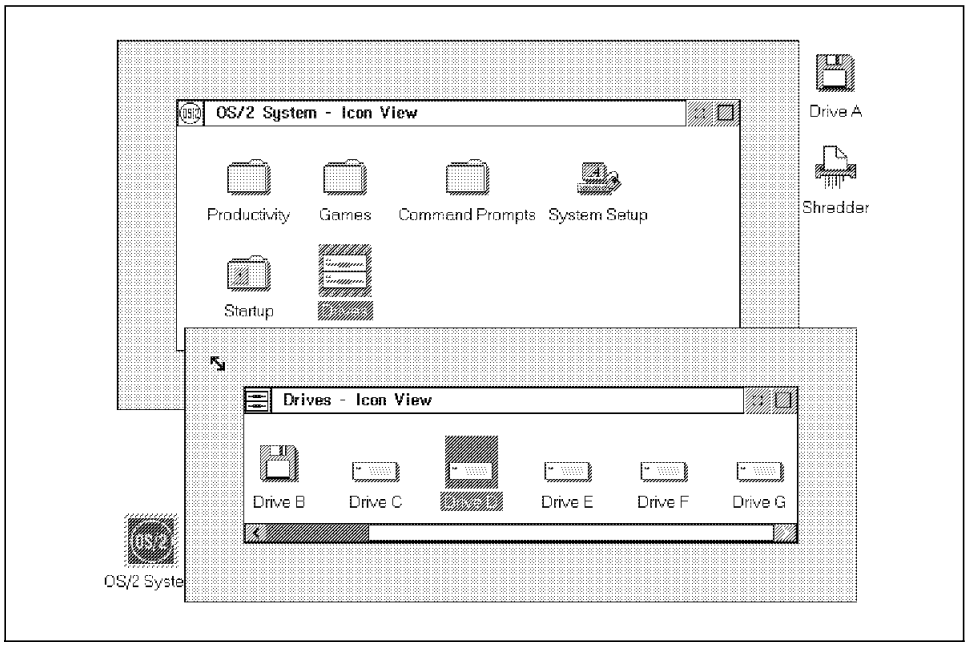


Figure 21. Desktop With a Border Width of 50

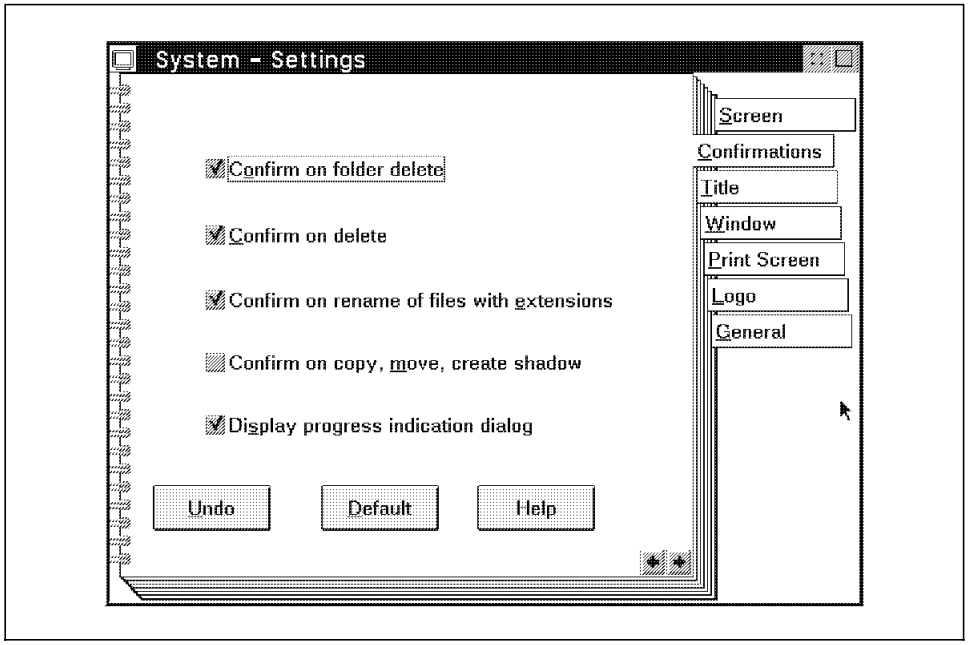


Figure 22. The Confirmations Page in the System Setting Notebook

Table 2. Confirmation Settings for PM_ControlPanel

Key	Value	Description
ConfirmCopyMoveEct	1	Confirmation of copy, move or shadow operation on an object.
	0	No confirmation of copy, move or shadow operation on an object.
ConfirmDelete	1	Confirmation of the deletion of an object.
	0	No confirmation of the deletion of an object.
ConfirmRenameFilesWithExt	1	Confirmation of the rename of a file with an extension that changes the extensions.
	0	No confirmation of the rename of a file with an extension that changes the extensions.
ConfirmSubDelete	1	Confirms the deletion of a folder that contains other folders.
	0	No confirmation the deletion of a folder that contains other folders.
DisplayProgressInd	1	Display the progress indication dialog box of a copy, move or delete progress.
	0	Does not display the progress indication dialog box of a copy, move or delete progress.

The other PM_ControlPanel values are listed in Table 2 and Table 3 on page 33. These setting are found in the System settings notebook in the System Setup Folder. These settings control global changes in the desktop and Workplace Shell. Some setting such as minimize of windows can be changed using either the PM_ControlPanel or the PM_DefaultSetup Application value. (See Table 4 on page 34 for the PM_DefaultSetup values) The PM_ControlPanel controls the confirmation windows, for example the confirmation window when you replace an existing object. Keyboard and cursor settings can be changed using PM_ControlPanel, and we can disable the print screen function.

Table 3. Other Settings for PM_ControlPanel Application		
Key	**Value**	**Description**
Beep	1	System beeps are enabled.
	0	System beeps are disabled.
BorderWidth	value	0 is no border and it can be set to any size.
CursorBlinkRate	0..890	0 is the fastest cursor blink rate and 890 the slowest cursor blink rate.
KeyRepeatDelay	0..890	0 is the minimum repeat delay and 890 is the maximum repeat delay.
KeyRepeatRate	1..20	Minimum key repeat delay is 1 and 20 is the maximum key repeat delay.
HiddenMinWindows	1	Hide window.
	2	Minimize window to viewer.
	3	Minimize window to desktop.
LogoDisplayTime	-1	Indefinite logo display.
	0	No logo display.
	value	Time to display the logo in milliseconds.
MinButtonType	1	Hide button.
	2	Minimize button.
NameClash	2	Auto-rename object.
	8	Replace existing object.
	16	Prompt for appropriate action.
PrintScreen	0	Print screen is disabled.
	1	Print screen is enabled.

DefaultSetup: There are two Application values that are not listed in the INI.RC file. These are PM_DefaultSetup and PM_Workplace. PM_DefaultSetup handles settings that affect the desktop globally. These are settings such as minimize to desktop or to the Minimized viewer, creation of group folders, such as in OS/2 1.3 and what the icon view and tree view should look like.

Table 4 on page 34 has a list of settings that can be used with PM_DefaultSetup. The Value, <OBJECTID> is a variable and can refer to any object ID, for example <WP_DESKTOP> which is the desktop.

Figure 23 on page 34 has examples of PM_DefaultSetup settings. The first line sets all windows to minimize to the desktop. The second line changes the view of folders that open with icon view. The icons in all the folders will be flowed with small icons.

```
"PM_DefaultSetup" "MINWIN" "DESKTOP"
"PM_DefaultSetup" "ICONVIEW" "FLOWED,MINI"
```

Figure 23. Examples of PM_DefaultSetup

Table 4. Settings for PM_DefaultSetup

Key	Value	Description
ACTIVEDESKTOP	<OBJECTID>	Identify the default desktop that is active
GROUPFOLDER	<OBJECTID>	Creates group folders similar to the groups in OS/2 1.3 and Windows
GROUPVIEW	ICONVIEW= FLOWED, NORMAL,...	These are the settings for the view of group folders. They use the same values as ICONVIEW which are listed in Table 6 on page 53
ICONVIEW	NONFLOWED, MINI,...	Use the same values as ICONVIEW which are listed in Table 6 on page 53
TREEVIEW	NORMAL, NOLINES,...	Use the same values as TREEVIEW which are listed in Table 6 on page 53
OPEN	<OBJECTID, OBJECTID,...>	Automatically opens the objects listed in the Value field. More than one object ID can be listed separated by commas.
MINWIN	DESKTOP	Minimize all windows to the desktop
	HIDE	Hide all windows.
HIDEBUTTON	YES	Change the buttons on windows to hide buttons. In some windows the button will not change but the window will hide.

Workplace: The other setting we found that we thought might be useful to a number of users is to set the global option to create a new window each time an icon is clicked. This is done using the application PM_Workplace and the key CCVIEW. This key has only the value of ON in the Value parameter. Any

other characters in the Value parameter will switch this value off. We suggest using OFF in the Value parameter, to switch the setting off, so it is clear what you are doing.

```
"PM_Workplace" "CCVIEW" "ON"
```

Figure 24. Setting Create New Window to be System Default

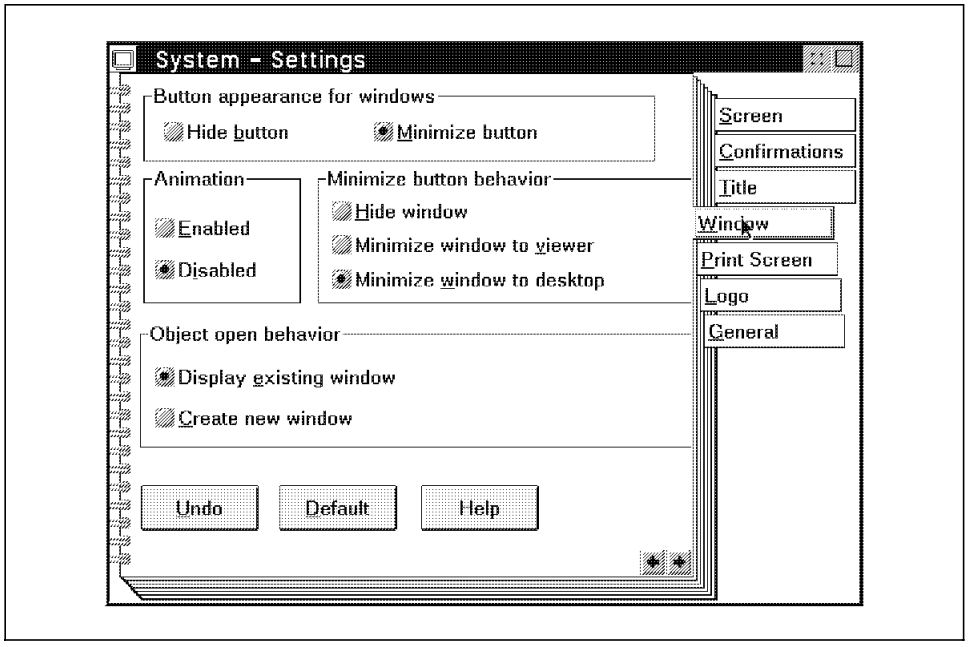


Figure 25. The Windows Page in the System Settings Notebook

3.5.2.2 Association Types

The next part of the INI.RC file is where Association Types are defined. This is a list of the default associations.

```
"PM_ASSOC_TYPE"      "Plain Text"       ""
"PM_ASSOC_TYPE"      "OS/2 Command File"       ""
"PM_ASSOC_TYPE"      "DOS Command File"       ""
"PM_ASSOC_TYPE"      "Executable"            ""
"PM_ASSOC_TYPE"      "Metafile"            ""
"PM_ASSOC_TYPE"      "PIF file"            ""
"PM_ASSOC_TYPE"      "Bitmap"            ""
"PM_ASSOC_TYPE"      "Icon"            ""
"PM_ASSOC_TYPE"      "Binary Data"            ""
"PM_ASSOC_TYPE"      "Dynamic Link Library"            ""
"PM_ASSOC_TYPE"      "C Code"            ""
"PM_ASSOC_TYPE"      "Pascal Code"            ""
"PM_ASSOC_TYPE"      "BASIC Code"            ""
"PM_ASSOC_TYPE"      "COBOL Code"            ""
"PM_ASSOC_TYPE"      "FORTRAN Code"            ""
"PM_ASSOC_TYPE"      "Assembler Code"            ""
"PM_ASSOC_TYPE"      "Resource File"            ""
"PM_ASSOC_TYPE"      "Printer-specific"            ""
```

Figure 26. Resource File Associations

You may change these associations or add your own associations. You can use these new associations to personalize your files and the links between data files and programs. For example, you can create an association with your name, associate a text file and then change the settings of your favorite editor to open the files with the association of your name.

Only the Applications and Keys are used to setup associations. The Value parameter is empty for associations. See Appendix B, “Associating Your Data Files” on page 235 for more information on associations.

```
"PM_ASSOC_TYPE"      "Mary-Ann"            ""
```

Figure 27. Adding Your Own Associations

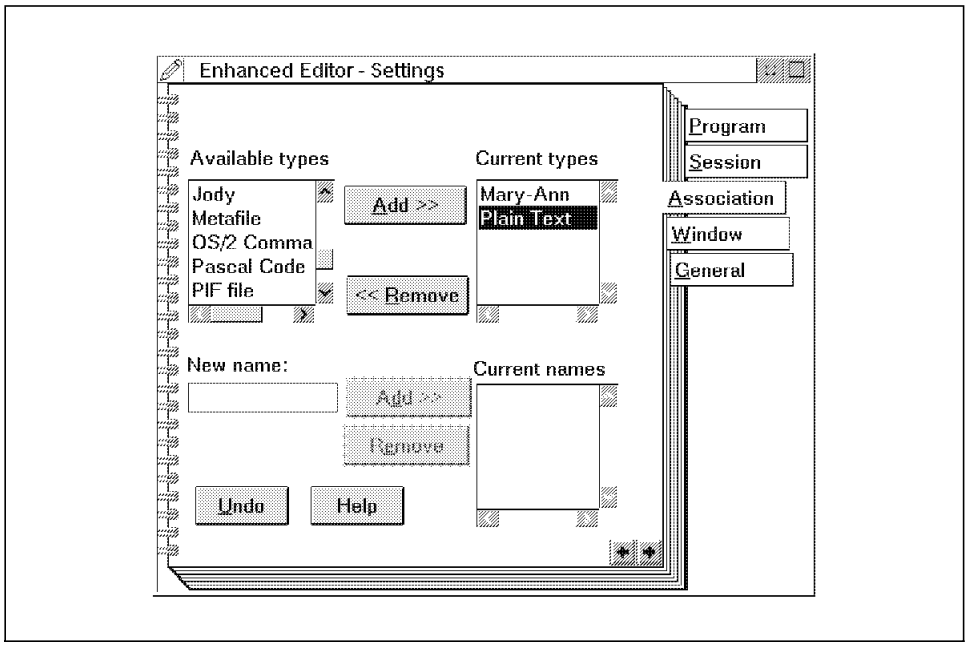


Figure 28. Using a New Association

3.5.2.3 Printer Driver and Font Driver Settings

The next part contains the printer driver and font driver settings. The Printer Device Drivers are defined here, along with the setup for the fonts. Even though you can add your own fonts and Printer Device Drivers in this list they will not be installed through this process. All that we are doing in adding them to this list is to notify the .INI file that they exist. If you wish to install printer drivers or fonts this way then their respective files must be installed prior to compiling the.INI file. If you do add another printer driver in the user .RC file you also need to update the PM_Spooler in the system .INI file. This is explained in a example later in this chapter. If you are creating a new desktop and don't add the printer drivers that are currently installed on your system, to the .RC file, you will have to reinstall the drivers. That is the only other way of getting the Printer driver objects back in the Workplace Shell.

```
"PM_DEVICE_DRIVERS"  "IBMNULL"

"C:\OS2\DLL\IBMNULL\IBMNULL.DRV"
"PM_Font_Drivers"    "PMATM"              "\OS2\DLL\PMATM.DLL"
"PM_Fonts"           "SYSMONO"            "\OS2\DLL\SYSMONO.FON"
"PM_Fonts"           "COURIER"            "\OS2\DLL\COURIER.FON"
"PM_Fonts"           "HELV"               "\OS2\DLL\HELV.FON"
"PM_Fonts"           "TIMES"              "\OS2\DLL\TIMES.FON"
"PM_Fonts"           "COURIERI"           "\OS2\DLL\COURIERI.FON"
"PM_Fonts"           "HELVI"              "\OS2\DLL\HELVI.FON"
"PM_Fonts"           "TIMESI"             "\OS2\DLL\TIMESI.FON"
"PM_Fonts"           "MARKSYM.OFM"        "\PSFONTS\MARKSYM.OFM"
"PM_Fonts"           "HELV.OFM"           "\PSFONTS\HELV.OFM"
"PM_Fonts"           "HELVB.OFM"          "\PSFONTS\HELVB.OFM"
"PM_Fonts"           "HELVBI.OFM"         "\PSFONTS\HELVBI.OFM"
"PM_Fonts"           "HELVI.OFM"          "\PSFONTS\HELVI.OFM"
"PM_Fonts"           "COUR.OFM"           "\PSFONTS\COUR.OFM"
"PM_Fonts"           "COURB.OFM"          "\PSFONTS\COURB.OFM"
"PM_Fonts"           "COURBI.OFM"         "\PSFONTS\COURBI.OFM"
"PM_Fonts"           "COURI.OFM"          "\PSFONTS\COURI.OFM"
"PM_Fonts"           "SYMB.OFM"           "\PSFONTS\SYMB.OFM"
"PM_Fonts"           "TNR.OFM"            "\PSFONTS\TNR.OFM"
"PM_Fonts"           "TNRB.OFM"           "\PSFONTS\TNRB.OFM"
"PM_Fonts"           "TNRBI.OFM"          "\PSFONTS\TNRBI.OFM"
"PM_Fonts"           "TNRI.OFM"           "\PSFONTS\TNRI.OFM"
```

Figure 29. RC File Printer Device Driver and Fonts

The application PM_Fonts can be used to register new fonts in the user .INI file. The font files must be installed on the hard disk and must be in the correct format. OS/2 converts Adobe Type font metrics to another format with an extension .OFM. If the .OFM file does not exist in the \PSFONTS directory then the relative line in the .RC file will be ignored. The .PFB files must also be in the \PSFONTS directory for printing purposes.

We were sucessful in installing additional fonts by:

1. Installing the desired font on an installed OS/2 system
2. Copying the .OFM and .PFB file from the /PSFONTS directory
3. Copying those files to the installing machine using User Exits during CID installation

The new font was then listed with the other fonts in the Edit Font dialog box in the Font Palette program and was also available for applications. The new

font can also be used for icon text in folders using the Value parameter ICONFONT (See Table 6 on page 53).

```
"PM_Fonts"          "BPRG____.OFM"        "\PSFONTS\BPRG____.OFM"
"PM_Fonts"          "BPB_____.OFM"        "\PSFONTS\BPB_____.OFM"
"PM_Fonts"          "BPBI____.OFM"        "\PSFONTS\BPBI____.OFM"
"PM_Fonts"          "BPI_____.OFM"        "\PSFONTS\BPI_____.OFM"
```

Figure 30. Installing Berthold Bodoni as Additional Fonts Using an .RC File

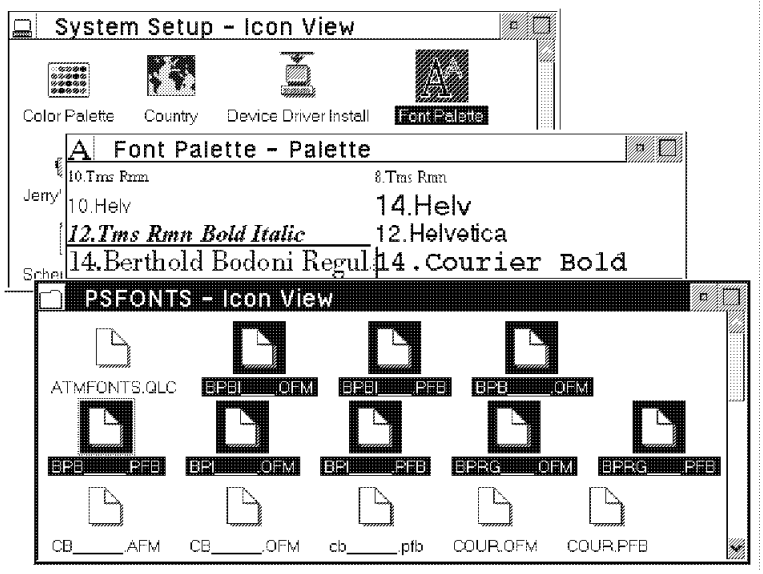


Figure 31. System After Copying Font Files and Compiling .RC File

3.5.2.4 System Settings

The OS/2 Workplace Shell settings follow after the fonts. These lines are used by the OS/2 system and we advise against changing or removing these lines from the.INI resource file. These lines contain information about the OS/2 version, system DLLs and printer spooler.

```
"PM_INFO"           "Version"           "2.1"
"PM_IMAGECNV"       "IMAGECNVPATH"      "\OS2\IMAGECNV"
"SYS_DLLS"          "LoadOneTime"       "REXXINIT"
"SYS_DLLS"          "LoadPerProcess"    "PMCTLS"
"PM_SPOOLER"        "QUEUE"             "LPT1Q;"
"PM_SPOOLER"        "PRINTER"           "PRINTER1;"
```

Figure 32. Resource File OS/2 System Settings

3.5.3 The International Information

The international settings are all the values contained in the Country Settings in the System Setup folder. You can only change the values in the Value parameter. You cannot add any new items to this list as it is fixed in the Workplace Shell.

```
"PM_National"  "iCountry"  "1"  /* Country code (phone ID of country)*/
"PM_National"  "iDate"     "0"  /* Date mode (0:MDY, 1:DMY, 2:YMD)   */
"PM_National"  "iCurrency" "0"  /* Currency mode 0: prefix, no sep   */
                                /*    1: suffix, no separation       */
                                /*    2: prefix, 1 CHAR separation   */
                                /*    3: suffix, 1 CHAR separation   */
"PM_National"  "iDigits"   "2"  /* Signif Decimal digits in Currency */
"PM_National"  "iTime"     "0"  /* time mode (0=12 hours clock, 1=24)*/
"PM_National"  "iLzero"    "0"  /* Leading zeros (0: no, 1: yes)     */
"PM_National"  "s1159"     "AM" /* Trailing string 0:00 to 11:59     */
"PM_National"  "s2359"     "PM" /* Trailing string 12:00 to 23:59    */
"PM_National"  "sCurrency" "$"  /* Currency Symbol string            */
"PM_National"  "sThousand" ","  /* Thousands separator string        */
"PM_National"  "sDecimal"  "."  /* Decimal separator string          */
"PM_National"  "sDate"     "-"  /* Date separator string             */
"PM_National"  "sTime"     ":"  /* time separator string             */
"PM_National"  "sList"     ","  /* List separator string             */
"PM_National" "iMeasurement"  "1"  /* 1=English, 2=Metric,
                                      3=Points, 4=Pica              */
```

Figure 33. Resource File PM International Setting

For example, to change the settings from American to European we would change the Country code, the Date mode, the Time Mode, Currency Symbol, Date separator string, and the Measurement.

In Figure 34 on page 41 we changed the National settings from the United States settings, which was default on the systems we were testing on, to the National settings for the Netherlands. The resulting desktop is shown in Figure 35 on page 41.

```
"PM_National"  "iCountry"  "031" /* Country code (country phone ID)   */
"PM_National"  "iDate"     "1"   /* Date mode (0:MDY, 1:DMY, 2:YMD)   */
"PM_National"  "iTime"     "1"   /* time mode (0=12 hours clock, 1=24)*/
"PM_National"  "sCurrency" "ƒ"       /* Currency Symbol string */
"PM_National"  "sDate"     "/"              /* Date separator string  */
"PM_National"  "iMeasurement"  "2"          /* 1=English, 2=Metric,
                                               3=Points, 4=Pica       */
```

Figure 34. Changing the PM National Settings

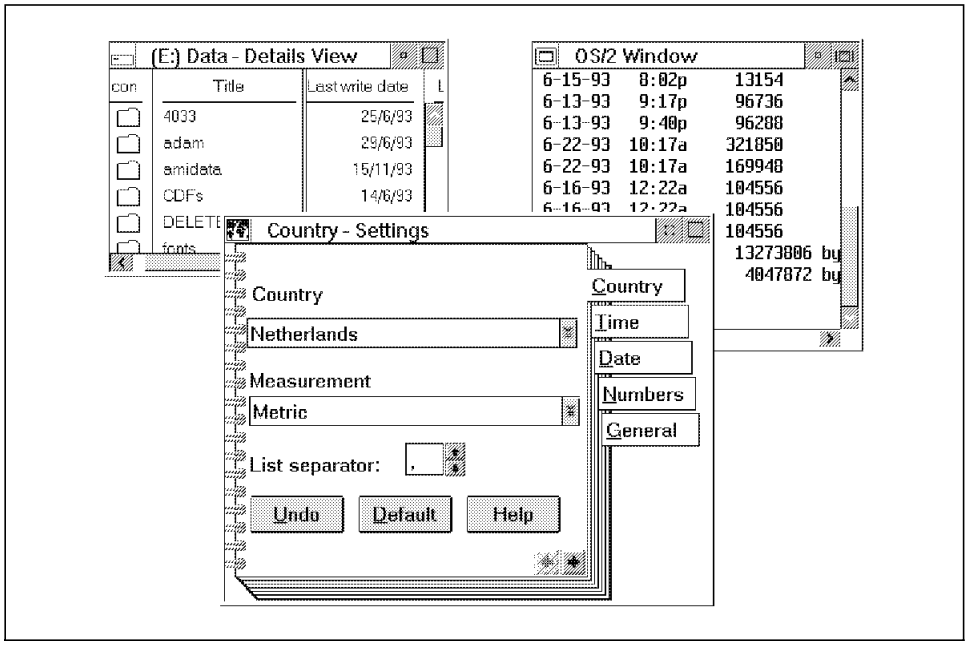


Figure 35. PM and Command Prompt Country and Date Settings Using an .RC File

Notes

The .RC international settings are for PM only! As you see in Figure 35, the date format in the PM Details view of a directory listing is in the new format but, the format in the OS/2 Window has remained unchanged. If you want to make a system wide global change then you must modify the COUNTRY=xxx,C:\OS2\SYSTEM\COUNTRY.SYS line in the CONFIG.SYS and reboot the system.

There is a comment in the INI.RC file that the PM National settings are only temporarily back in the .RC files. The International settings might not be in the .RC files in future releases of OS/2.

3.5.4 Desktop Settings

The last part of this area is where we setup the desktop. This includes the:

- Settings for the 8514 adapters
- Paths to other .INI files
- Settings for WIN-OS2
- Install group folders, such as the OS/2 Version 1.3 Main Group folder
- Running the OS/2 Tutorial the first time you load the Desktop
- The installation of the Drive A object
- The installation of the Desktop and the Start Here object
- Removing the PMDDE object from the Workplace

All of these settings can be changed. If you have programs that use their own .INI files you can add them to the.INI files by including them in this list. You have to know the programs′s name, the key it uses in the user .INI file to refer to the program′s .INI file, and the path of the program′s .INI file. The settings for EPM are as follows: EPM is the program name, EPMIniPath is the key and the path of the.INI file is ?OS2EPM.INI.

The “WIN-OS2” line is the global WIN-OS2 settings for Windows 3.1. These are the default settings. If you change these settings it will be a global change and all the Window applications will have these settings. It can be changed after you have installed the Workplace Shell by changing the values in the WIN-OS2 Setup settings icon found in the System Setup folder.

The Drive object can be changed to be a shadow of any drive on your system. This is done by making copies of the line:

"PM_Workplace:InstallDiskOnDesktop" "**A**" "ICONPOS=80 8"

You can change the drive letter from "A" to "C" to have a shadow of Drive C instead of or in addition to the Drive A on your desktop. For example:

"PM_Workplace:InstallDiskOnDesktop" "**C**" "ICONPOS=70 8"

```
"PM_IBMBGA"    "ALTSYSFONT" "0"  /* For 8514 adapter only */
"PM_IBMBGA"    "FASTSS"     "0"  /* For 8514 adapter only */

"EPM"          "EPMIniPath" "\OS2\EPM.INI"
"PMDiary"      "IniPath"    "\OS2\PMDIARY.INI"

/* Change default for Win-OS/2 Setup Object - R206 60555 */
"WINOS2"       "PM_GlobalWindows31Settings" "DPMI_MEMORY_LIMIT=64;
     PROGTYPE=PROG_31_STD;KBD_ALTHOME_BYPASS=1;
     VIDEO_SWITCH_NOTIFICATION=1;VIDEO_8514A_XGA_IOTRAP=0"

"PM_Workplace:InstallGroups" "1" "1"
"PM_Workplace:InstallAutorun" "OS/2 Tutorial"
     "EXENAME=TUTORIAL.EXE,STARTUPDIR=\OS2\HELP"
"PM_Workplace:InstallDiskOnDesktop" "A" "ICONPOS=80 8"

"PM_InstallObject" "Desktop;WPDesktop;?:\" "OBJECTID=<WP_DESKTOP>"
"PM_InstallObject" "Start Here;WPProgram;<WP_DESKTOP>"
     "EXENAME=STHR.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\HELP;
      HELPPANEL=9278;OBJECTID=<WP_STHR>"

/* remove PMDDE object from workplace */
"PM_InstallObject" "Deleted;WPProgram;<WP_TOOLS>;DELETE"
     "OBJECTID=<WP_PMDDE>"
```

Figure 36. Resource File Desktop Settings

You can also change the desktop name to something other than "Desktop." This new name will be displayed in your Window List. The desktop directory will have the same name as the desktop for HPFS file systems and the normal abbreviated name for FAT file systems. For example, to add a new desktop called "My New Desktop" we change the install desktop line to:

"PM_InstallObject" "**My New Desktop;**WPDesktop;?:\" "OBJECTID=<WP_DESKTOP>"

On a FAT file system this creates a new desktop directory named C:MY_NEW_D and the desktop named "My New Desktop." The new desktop

name can be seen when you open the Task List window as well as in the directory structure of the boot drive, as can be seen in Figure 37 on page 44.

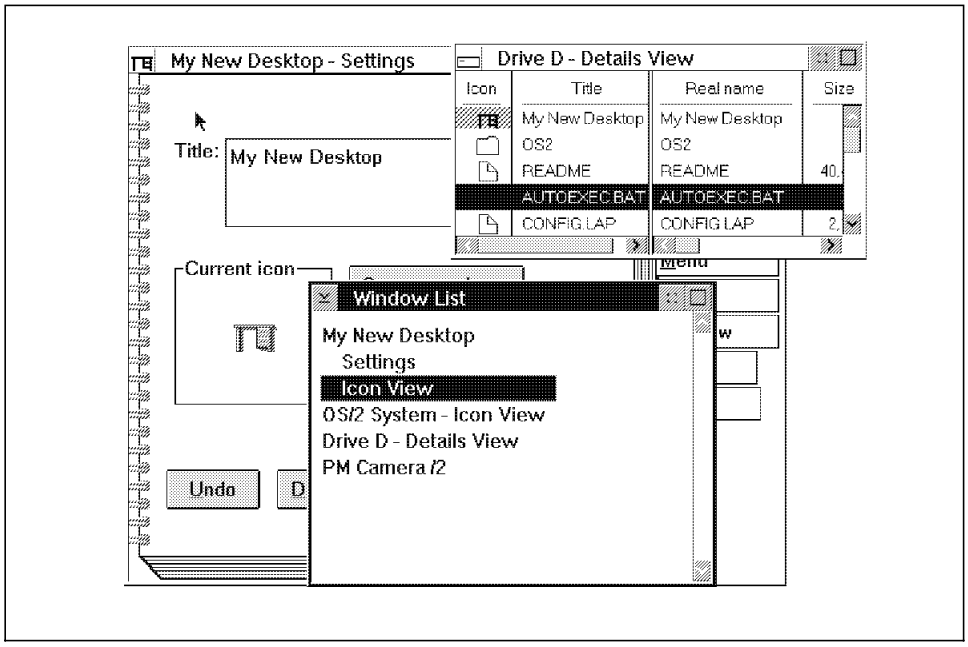


Figure 37. My New Desktop

3.5.5 Object Definitions

The main objective of being able to manipulate the desktop is to create, delete, and change objects on the desktop. For this we will make use of the application PM_InstallObject. This Application value is used to create the desktop and all the objects on the desktop such as folders, programs, the shredder, and more. Some of the features that you can set here that you can not set on the desktop are being able to define whether objects can or cannot be deleted, moved, copied, or dragged.

In 3.4, “The Structure of .RC File Commands” on page 26 we discussed the .RC file command line structure:

```
"Application" "Key" "Value"
```

These will now be explained as they relate to object definitions.

3.5.5.1 The Application Parameter

To make any changes to the objects on the desktop we use one Application called **PM_InstallObject**. The name can be misleading as PM_InstallObject can also be used to change the appearance of existing objects and delete existing objects.

The Application parameter's value will always be PM_InstallObject and the Key and Value parameter are used to tell the system the identity of the object. The Key parameter is used to specify the type of change and the Value parameter is used to add certain settings to the object.

```
"PM_InstallObject""Calculator;WPProgram;<WP_TOOLS>;UPDATE"
  "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\APPS;
   HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 38. The Application Parameter PM_InstallObject

3.5.5.2 The Key Parameter

The Key parameter is where we specify the type of object, where the object is located, or where it should be located if it is created. The Key parameter is also used to associate the object to a Workplace Shell class and give the object a title.

The Key parameter is divided into three statements with an option of a fourth part. The statements are separated with a semicolon and must always be in the same order. These four statements, in order, are:

1. Title of the object
2. Workplace Shell Class of the object
3. Location of the object
4. An option to perform a function. These options are:
   - REPLACE
   - FAIL
   - UPDATE
   - DELETE

```
"PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
  "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\APPS;
   HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 39. The Key Parameter of the Application PM_InstallObject

The Title of the Object: The title of the object must be unique in the folder where the object will be placed. The object will not be referred to by its title but by the OBJECTID and this has to be unique for the whole Workplace Shell. An OBJECTID is not necessary but it makes it easier to refer to the objects. The title of the object can be any alphanumeric character that is allowed by the desktop and can be up to 256 characters.

```
"PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
  "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\APPS;
   HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 40. The Title of the Object of the Key Parameter

The Class of the Object: The class of the object determines the type of object we are creating. Most of the objects are class WPDataFile, WPProgram or WPFolder. This is a data file, program, or folder, respectively. Using these classes we can create shadows of any object, or any of the objects, in the Templates folder.

The WPDataFile class contains most of the data file variations, such as bitmaps and icons. WPFolder class has the different types of folders. Some of these folders will have objects in them without any specification from the user. Examples of these folders are the WPDrives which will have all the drives and WPTemplates which will have all the templates. WPAbstract class are the objects in the System Setup folder. The WPTransient class objects are used by the system and have little use for the user. Figure 42 on page 47 is a list of the classes under the Workplace Shell.

Note

WPProgramFile and **WPCommandFile** have the same properties as **WPProgram**. We suggest you use WPProgram to install program objects.

```
"PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
  "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\APPS;
   HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 41. The Class of the Object of the Key Parameter

```
WPObject
  WPFileSystem          WPAbstract            WPTransient
    WPDataFile            WPClock               WPCnrView
      WPBitmap            WPCountry               WPDiskCV
      WPIcon              WPDisk                  WPFolderCV
      WPPointer           WPKeyboard            WPFilter
      WPProgramFile       WPMouse                 WPFinder
        WPCommandFile     WPPalette             WPMinWindow
      WPMet                 WPSchemePalette     WPJob
      WPPif                 WPColorPalette      WPPort
    WPFolder                WPFontPalette       WPPrinterDriver
      WPDesktop           WPProgram             WPQueueDriver
      WPStartup           WPPrinter
      WPDrives              WPRPrinter
      WPMinWinViewer      WPShadow
      WPFindFolder          WPNetLink
      WPNetgrp            WPShredder
      WPNetwork           WPSound
      WPServer            WPSpecialNeeds
      WPSharedDir         WPSpool
      WPTemplates         WPSystem
      WPRootFolder        WPPower
```

Figure 42. The Object Class List

The Location of the Object: The locations listed in this table are created during the installation of OS/2. You can use these locations to add your own objects. For example, you can add shadows of objects to the Startup folder named **<WP_START>.** These objects will be in the Startup folder and start when the Workplace Shell is opened. Other common folders created by the system when you install other programs are also listed. For example, the Mahjongg folder is created when you install the Mahjongg game. The installation of this game is documented in your README file in the root directory of your boot drive. The Multimedia folders are created when you install the Multimedia disks supplied with OS/2, and the Toolkit when you

install the Toolkit software. When you add your own folders the OBJECTID you give the folder can be used as a location for your objects.

```
"PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
  "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\APPS;
   HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 43. The Location of the Object of the Key Parameter

Table 5. Predefined Object IDs

Object ID	**Description**
<WP_DESKTOP>	Desktop
<WP_START>	Startup folder
<WP_OS2SYS>	OS/2 System folder
<WP_TEMPS>	Templates folder
<WP_CONFIG>	System Setup folder
<WP_INFO>	Information folder
<WP_DRIVES>	Drives folder
<WP_NOWHERE>	Hidden folder
<WP_PROMPTS>	Command Prompts folder
<WP_TOOLS>	Productivity folder
<WP_GAMES>	Games folder
Some other common OS/2 folder objects created by the system	
<MAH_FOLDER>	Mahjongg folder
<MMPM2_FOLDER>	Multimedia folder
<MMPM2_SOUNDS>	Multimedia Sound Bites folder
<MMPM2_MOVIES>	Multimedia Movies folder
<TK_TOOLKIT>	Toolkit 2.1 folder
<TK_DEVINFO>	Toolkit Information folder
<TK_CSAMPLE>	Sample Programs folder

The Option of an Object: The options are the functions we are using on the objects. If no option is used, and the object does not exist, a new object will be created. If the object does exist then nothing will be done unless you specify an option!

If you are changing an existing object you must specify the title, location and class of the object. In the Value parameter you need to have the OBJECTID and the fields you want to change.

Option	Description
REPLACE	Delete the old object and create a new object
FAIL	If the object already exists it will not do anything to that object This is when you want to create new objects but do not want to override existing ones
UPDATE	Updates an object
DELETE	Deletes an object

```
"PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
  "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\APPS;
   HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 44. The Option of the Object of the Key Parameter

3.5.5.3 The Value Parameter

The Value parameter is used by the Application PM_InstallObject to determine the settings of the object. The Value parameter is a series of "keyname=value" settings, separated by semicolons, that determine the behavior of the object. Each class has its own set of keyname=value settings for the Value parameter. We have found that by looking at the Settings of the objects created by the classes gives some idea of what values can be used. For example, although the System Clock is a PM program it belongs to the class WPClock. When you look at the System Clock settings there is no "Path and filename" entry field. Therefore you cannot use the entry "EXENAME=" in the Value parameter for the class WPClock. The System Editor is of class WPProgram and when you open the Settings for the System Editor you will find the "Path and Filename" entry field and therefore you can use the "EXENAME=" value for the class WPProgram.

We will be discussing the three more popular classes, Folders, Programs and Shadows. These classes have most of the settings that you will find in the other classes.

```
"PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
     "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;
           STARTUPDIR=\OS2\APPS;
           HELPPANEL=20272;OBJECTID=<WP_DCALC>"
```

Figure 45. The Value Parameter of the Object

Note

The tables with Value parameter settings have the keyname in the first column, and the value in the second column. These are the combinations that are used in the Value parameter. The third column contains a short description of what each setting does.

Creating Folder Objects: To create a folder we use the WPFolder class. All the folders that are inherited from the WPFolder class can be used, but they all have specific functions. For example, the WPDrives folder is a folder that will have all the Drives in it when it is created. You can add your own objects to a specific folder that is a subclass of WPFolder.

You can name the folder any title, with characters that are permitted by the desktop. If the title exists at the location it will not be recreated, and will be changed or deleted if it was specified. The location of the new folder must exist in order to create it.

You can use UPDATE, REPLACE, FAIL, and DELETE on folders. If the folder contains any icons, the icons will be deleted during a REPLACE or DELETE. If the icons cannot be deleted, the folder will not be deleted or replaced.

Table 6 on page 53 contains the settings to specify the views of the folder. You can change the Icon view and Tree view of the folder but not the Details view. The only changes you can make to the Details view is the fields you want displayed; however, this cannot be done using the .RC files. The ICONVIEW and TREEVIEW settings can have a list of styles separated by commas.

```
"PM_InstallObject" "Folder;WPFolder;<WP_FOLDER>"
   "ICONVIEW=FLOWED,MINI;OBJECTID=FOLDER"
```

When you enter two values that are opposites of each other, for example FLOWED and NONFLOWED, the last value will be the setting of the window. The folder in the following example will be non-flowed.

```
"PM_InstallObject" "Folder;WPFolder;<WP_FOLDER>"
   "ICONVIEW=FLOWED,NONFLOWED;OBJECTID=FOLDER"
```

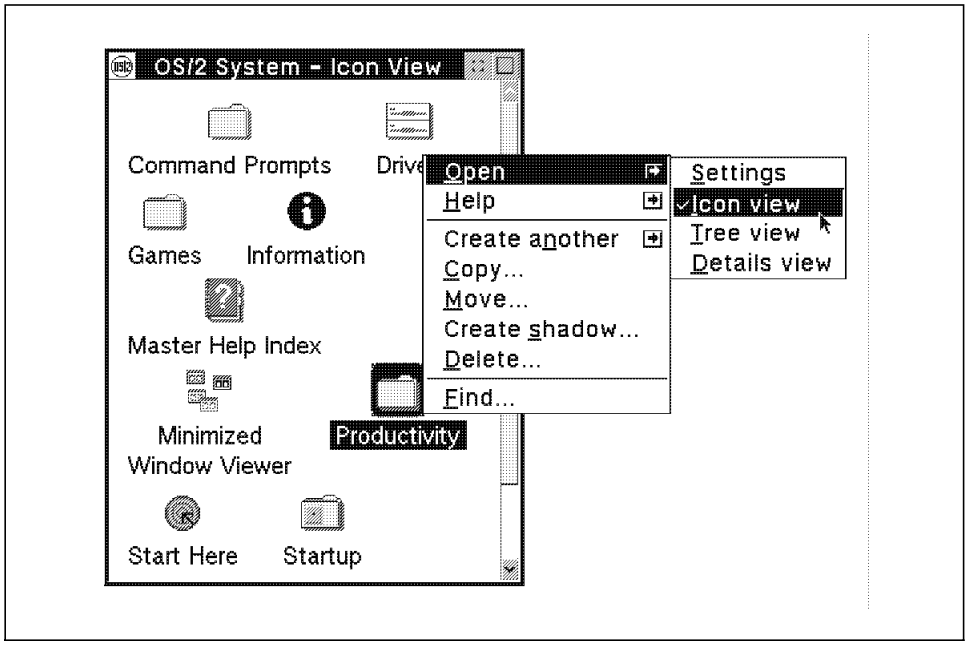


Figure 46. Open Options of a Folder

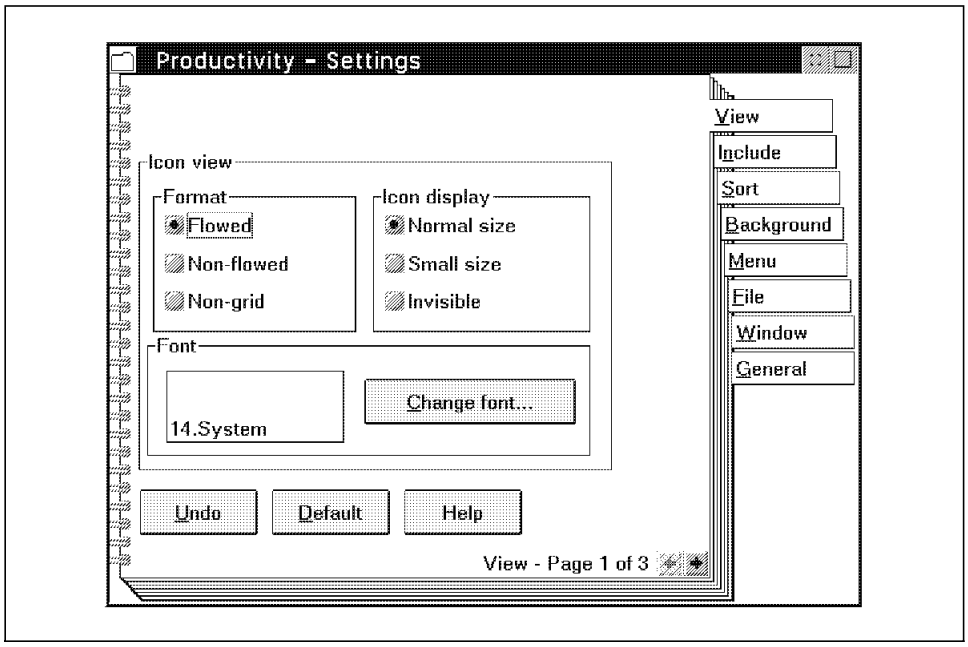


Figure 47. Icon View Page of a Folder Settings Notebook

Table 6. WPFolder View Setup String Parameters

Keyname	Value	Description
OPEN	ICON	Open icon view when object is created/updated.
	TREE	Open tree view when object is created/updated.
	DETAILS	Open details view when object is created/updated.
ICONVIEW	s1[,s2,...sn]	Set icon view to specified style(s).
	Styles for ICONVIEW:	
	FLOWED	flowed list items
	NONFLOWED	non-flowed list items
	NONGRID	non-gridded icon view
	NORMAL	normal size icons
	MINI	small icons
	INVISIBLE	no icons
TREEVIEW	s1[,s2,...sn]	Set tree view to specified style(s).
	Styles for TREEVIEW:	
	NORMAL	normal size icons
	MINI	small icons
	INVISIBLE	no icons
	LINES	lines in tree view
	NOLINES	no lines in tree view
ICONFONT	value	Font size and facename. See Font Notes following.
TREEFONT	value	Font size and facename. See Font Notes following
DETAILSFONT	value	Font size and facename. See Font Notes following

Font Notes

The format for the value is:

```
size.facename fontstyle
```

For example, code to change the Information Folder icon view font to Courier Bold:

```
ICONFONT=8.Courier Bold
```

To change to Courier Normal:

```
ICONFONT=8.Courier
```

To change to Courier Bold Italic:

```
TREEFONT=8.Courier Bold Italic
```

For a complete listing of fonts see Table 41 on page 173 but note that the double quotes (″ ″) used in the .INI files are not used in the value parameter of the .RC files.

Hint

If you are not sure what the string should look like you can do the following:

1. Create a folder and name it something simple, like MYFOLD
2. Change the font size/name using the Open/Settings/Change font button
3. Close the settings
4. From an OS/2 command line determine the name of your desktop

   For a typical 2.0 FAT file system it would be something like C:OS!2_2.0_D (It gets easier for 2.1 where the Desktop is normally named C:DESKTOP).
5. Locate the folder you created C:OS!2_2.0_DMYFOLD
6. Type EAUTIL C:OS!2_2.0_DMYFOLD MYFOLD.EAS /S /P
7. Press Enter

This will create a MYFOLD.EAS file. You can use a text program to view this file and you'll see the values required. Note that this is not a pure text file but the font information is in a readable form.

If you select a new bitmap background for your folder *and* the folder is set to open on start up then the background will not be the bitmap. When you open the settings for the folder you will find that the background is selected to the bitmap and *then* the folder's background will change. If the folder is *not* set to open on start up then the background will be the bitmap when you open it after start up. We have found similar results when we tried to change the backgrounds of open folders using REXX procedures.

Note

The filename of the bitmap must not be enclosed in quotes or brackets.

```
"PM_InstallObject" "Folder;WPFolder;<WP_FOLDER>"
"BACKGROUND=D:\BITMAPS\OS2.BMP;OBJECTID=FOLDER"
```

Figure 48. Setting a Bitmap Background for a Folder

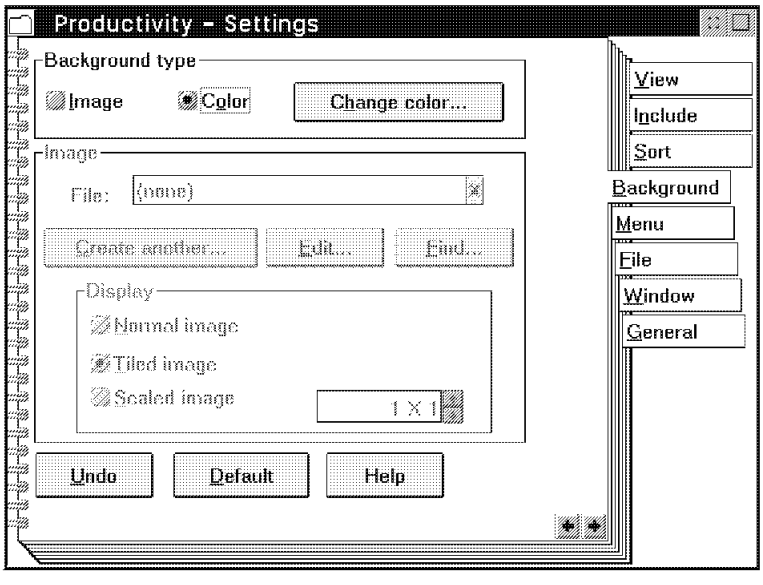


Figure 49. Background Page of a Folder Settings Notebook

Table 7. WPFolder Background Setup String Parameters

Keyname	Value	Description
BACKGROUND	filename	Sets the folder background. The value 'filename' is the name of a file in the OS2BITMAP directory of the boot drive.

Work area folders have two special features:

1. When you close the folder all the windows belonging to the objects within the folder are closed automatically. The view of each object is saved when is closed. When the folder is opened the next time, all the windows that were open when the folder was closed are displayed with their previous view.
2. When you hide the window of a work area folder all the open windows belonging to the objects in the folder are hidden automatically. When you show the work area folder the windows of the objects in the folder are displayed. If the work area folder is minimized only the work area folder's icon is displayed in the Minimized Window Viewer or on the desktop. The icons belonging to the objects in the work area folder are not displayed.

Note

The work area function only works with folders in the icon view or in the details view, but not with the tree view.

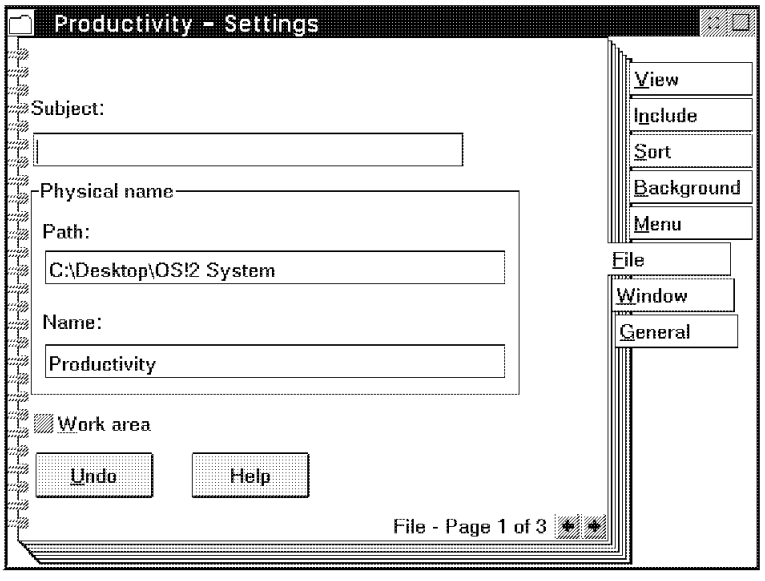


Figure 50. Work area Setting on the File Page of a Folder Settings Notebook

Table 8. WPFolder File Setup String Parameters		
Keyname	**Value**	**Description**
WORKAREA	YES	Make the folder a Work area folder
	NO	Make the folder a non-Work area folder.

Table 9 on page 59 shows the settings you can specify for the Window page of a folder settings. If you do select VIEWBUTTON=HIDE and have a MINWIN setting, the MINWIN setting will not be selected until you change the Button Appearance in the folder's settings to "Minimize Button." The CCVIEW indicates what the Object open behavior selection will be in the folder.

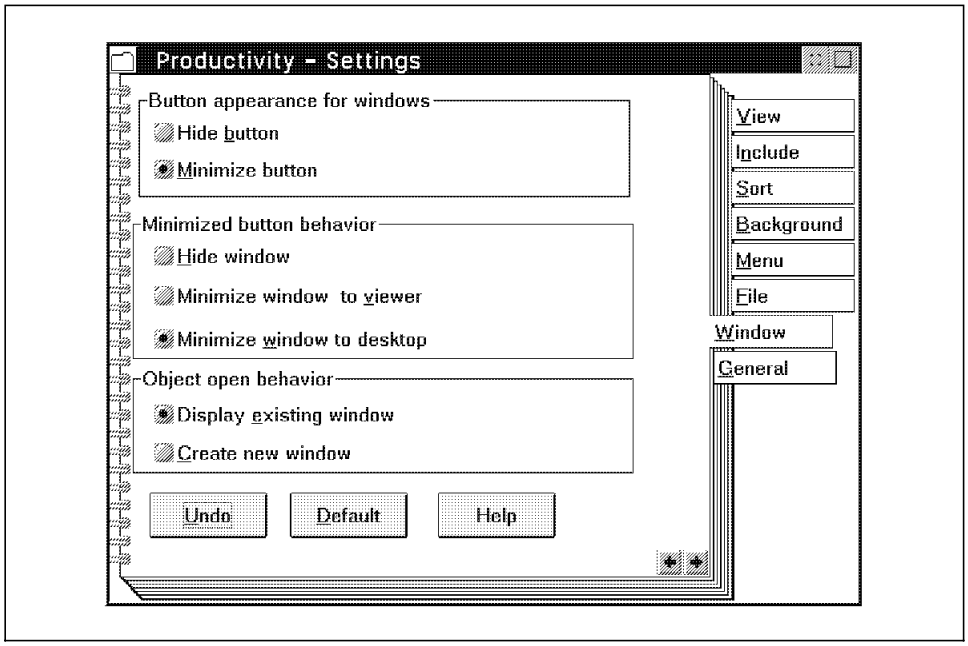


Figure 51. Window Page of a Folder Settings Notebook

Table 9. WPFolder Window Setup String Parameters

Keyname	Value	Description
MINWIN	HIDE	Views of this object will hide when their minimize button is selected.
	VIEWER	Views of this object will minimize to the minimized window viewer when their minimize button is selected.
	DESKTOP	Views of this object will minimize to the Desktop when their minimize button is selected.
VIEWBUTTON	HIDE	Views of this object will have a hide button as opposed to a minimize button.
	MINIMIZE	Views of this object will have a minimize button as opposed to a hide button.
CCVIEW	YES	New views of this object will be created every time the user selects open.
	NO	Open views of this object will resurface when the user selects open.

Table 10 shows the settings you can specify for the General page of a folder's settings.

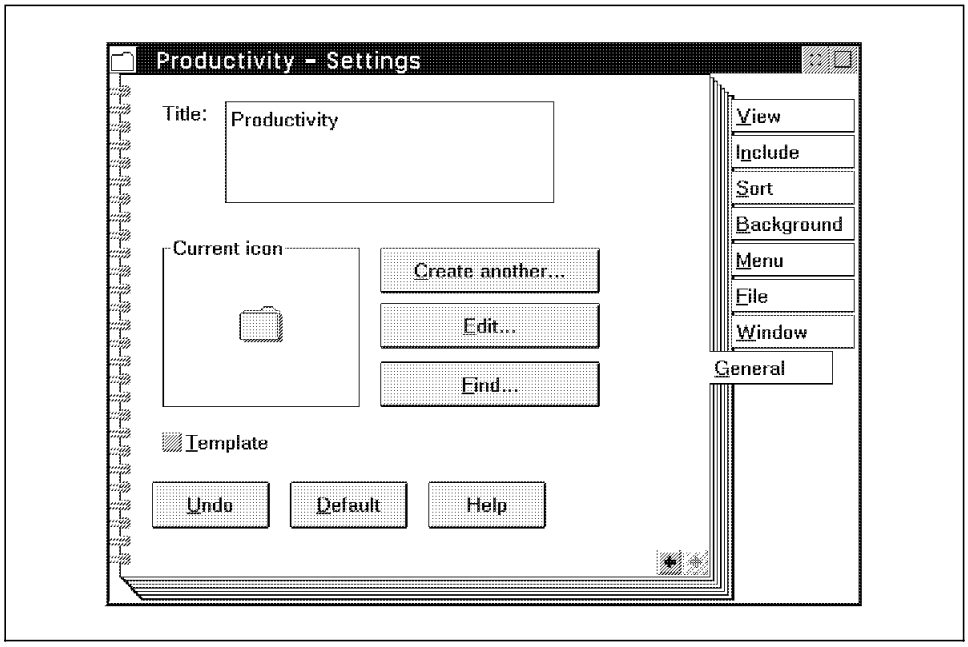


Figure 52. General Page of a Folder Settings Notebook

Table 10. WPFolder General Setup String Parameters

Keyname	Value	Description
TEMPLATE	YES	Makes object a template.
	NO	Resets objects template property.
TITLE	value	Used to assign a name/title to an object. (This is the Icon name!)

For the ICONFILE value, don't use quotes or brackets to enclose the filename of the icon file.

```
"PM_InstallObject" "Folder;WPFolder;<WP_FOLDER>"
   "ICONFILE=D:\ICONS\CALVIN.ICO;
    OBJECTID=FOLDER"
```

Figure 53. Set an Icon to a Folder Using ICONFILE

To use the ICONRESOURCE you need to know the icon's ID in the DLL file. The ICONPOS and ICONVIEWPOS are resolution independent. They work on a percentage value, where 0 0 is the bottom left corner and 100 100 is the top right corner of any display. This example places the folder icon in the middle of the screen, and opens the folder in the bottom right quarter of the screen.

```
"PM_InstallObject" "Folder;WPFolder;<WP_FOLDER>"
   "ICONPOS=50 50;ICONVIEWPOS=50 0 50 50;OBJECTID=FOLDER"
```

If you enclose the TITLE in quotes the quotes will be part of the new title. The HELPFILENAME has the same structure as the BACKGROUND and ICONFILE filenames and must not be enclosed in quotes or brackets. The last number of settings are all flags and determines the behavior of the folder icons and what is in the folder's system menu.

Warning

Be very cautious with settings such as NOTVISIBLE=ON as you would have very little control over the folder icon.

Table 11. WPFolder Icon Related Setup String Parameters

Keyname	Value	Description
ICONFILE	filename	This sets the object's icon.
ICONRESOURCE	id module	This sets the object's icon. 'id' is the identity of an icon resource in the 'module' dynamic link library (DLL). For example: `ICONRESOURCE=73 PMWP;` This would indicate resource 73 in PMWP.DLL.
ICONPOS	l% b%	This sets the object's initial icon position. The l and b values represent the position in the object's folder in percentage coordinates.
ICONVIEWPOS	l% b% w% h%	This sets the object's initial size. The values represent relative position in percentage coordinates. For example: ICONVIEWPOS=25 25 50 50 would create a folder whose bottom left corner is 25% from the left and 25% from the bottom and half the screen width/height.

Table 12. WPFolder Miscellaneous Setup String Parameters

Keyname	Value	Description
OBJECTID	<name>	This sets the object's identity. The object ID will stay with the object even if it is moved or renamed. An object ID is any unique string preceded with a '<' and terminated with a '>'. This may also be a real name specified as a fully qualified path name. IMPORTANT: For any object you create you should use a unique OBJECTID! Do this for two reasons: 1. If you use an object ID it signifies a unique object that will not be recreated if you use the "FailIfExists" flag in your REXX call. Not using an object ID would cause multiple objects to be created if the same program was run over and over. 2. Should you need to later delete it or change your object, you can use this object ID in your REXX call to refer to it. Also one should not use an object ID that starts with "WP_" as many OS/2 objects use those, consider those reserved characters.
HELPPANEL	id	This sets the object's default help panel.
HELPLIBRARY	filename	This sets the help library.
OPEN	SETTINGS	Open settings view of object when created/updated.
	DEFAULT	Open default view of object when created/updated. Don't forget for folder objects you can use OPEN with these values: ICON, TREE, DETAILS

Table 13. WPFolder Object Properties Setup String Parameters		
Keyname	**Value**	**Description**
NODELETE	YES	Will not allow you to delete the object.
	NO	Resets the object's no delete property.
NOCOPY	YES	Will not allow you to make a copy.
	NO	Resets the object's no copy property.
NOMOVE	YES	Will not allow you to move the object to another folder, will create shadow on a move.
	NO	Resets the object's no move property.
NODRAG	YES	Will not allow you to drag the object.
	NO	Resets the object's no drag property.
NOLINK	YES	Will not allow you to create a shadow link.
	NO	Resets the object's no link property.
NOSHADOW	YES	Will not allow you to create a shadow link.
	NO	Resets the object's no shadow property.
NORENAME	YES	Will not allow you to rename the object.
	NO	Resets the object's no rename property.
NOPRINT	YES	Will not allow you to print it.
	NO	Resets the object's no print property.
NOTVISIBLE	YES	Will not display the object.
	NO	Resets the object's not visible property.

Creating Program Objects: To create a program object we make use of the WPProgram class. There are two other classes that will create program objects as well. They are WPProgramFile and WPCommandFile. The difference between WPProgram and the other two classes is that the other two actually create a file on the disk while WPProgram only creates an object on the desktop. The name of the file created on the disk is the name of the object and not the name specified in the EXENAME parameter in the Value field. The reason for this is that WPProgramFile and WPCommandFile are subclasses of WPDataFile, which is actual physical files. WPProgram, on the other hand, is of the class WPAbstract, which is abstract objects on the desktop without any physical files. We recommend that you use the WPProgram class as it is the same as the Program template in the Templates folder.

You can title the program object any name using characters that are permitted by the desktop. If the title already exists at the location it will not be recreated but, instead, the existing object will be changed or deleted depending on your specifications. The location of the new program object must exist in order to create it.

You can use UPDATE, REPLACE, FAIL and DELETE on program objects. REPLACE and DELETE will return an error if the object exists and cannot be deleted. This is the case if the existing object's NODELETE flag is set to YES.

Table 14 contains the settings to specify the Program parameters of an object. The EXENAME value is the physical filename and full path of the program that this object should launch. You can use *?* to refer to the drive that OS/2 boots from. The filename must not be enclosed in quotes or brackets as these characters will then be part of the filename. This also applies to the STARTUPDIR value. In the STARTUPDIR value if you don't enter the drive letter, it will default to the drive where the program file resides.

```
"PM_InstallObject" "Testing;WPProgram;<WP_DESKTOP>"
   "EXENAME=?\UTILITY\TESTING.EXE;
    STARTUPDIR=\UTILITY;
    OBJECTID=TESTING"
```

Figure 54. EXENAME and STARTUPDIR for a Program Object

Table 14. WPProgram Program Setup String Parameters

Keyname	Value	Description
EXENAME	filename	Sets path and name of the program.
PARAMETERS	params	Sets the parameters list, which may include substitution characters. See the Programs Parameters Substitution characters table.
STARTUPDIR	pathname	Sets the working directory.

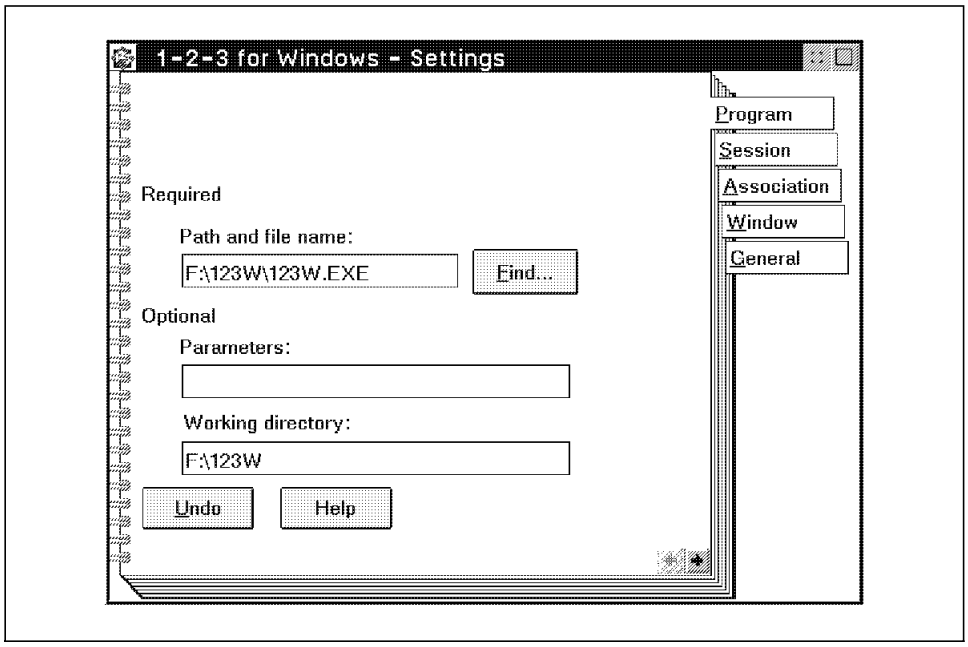


Figure 55. Program Page of a Program Settings Notebook

The values for the PARAMETERS settings are in Table 15 on page 67. If you press F1 (Help) while your cursor is in the Parameters field in the Program Settings notebook you will get the same information. Using the %* combination as a parameter will use the filename of the object dragged onto the program object.

```
"PM_InstallObject" "Testing;WPProgram;<WP_DESKTOP>"
   "EXENAME=?\UTILITY\TESTING.EXE;
    PARAMETERS=[ ]
    OBJECTID=TESTING"
```

Figure 56. Setting Parameters for a Program Object

Table 15. Program Parameters Substitution Characters

Parameter	Description
[ ]	You are prompted to type any parameters you want to use.
[text]	Characters placed inside of the brackets are displayed as the prompt string.
no parm	If the program object is started by clicking on it no parameters are passed. If you start the program object by dragging a file over it, the full filename is passed.
%	No parameters are passed. Useful for program objects you may want to start from a folders pop-up menu.
%*	Enables you to open a data file object in one of two ways. You can drag the data file object to the program object and drop it. Or, you can open a data file object that you associated to a program.
%**P	Insert drive and path information without the last backslash ().
%**D	Insert drive with ':' or UNC name.
%**N	Insert file name without extension.
%**F	Insert file name with extension.
%**E	Insert extension without leading dot. In HPFS, the extension always comes after the last dot.

The program sessions specifies the type of program that you are going to run. Table 17 on page 70 has the settings for the **PROGTYPE=** Value parameter. The settings for the OS/2 2.1 system session values in Table 17 on page 70 is the settings to setup Windows 3.1 programs.

When you use an * (asterisk) instead of a filename for EXENAME you will get a command prompt depending on your session type. This is set with the PROGTYPE value. This example has the settings for an OS/2 Window:

```
"PM_InstallObject" "OS/2 Window;WPProgram;<WP_DESKTOP>"
   "EXENAME=*;PROGTYPE=WINDOWABLEVIO;OBJECTID=OS2WIN"
```

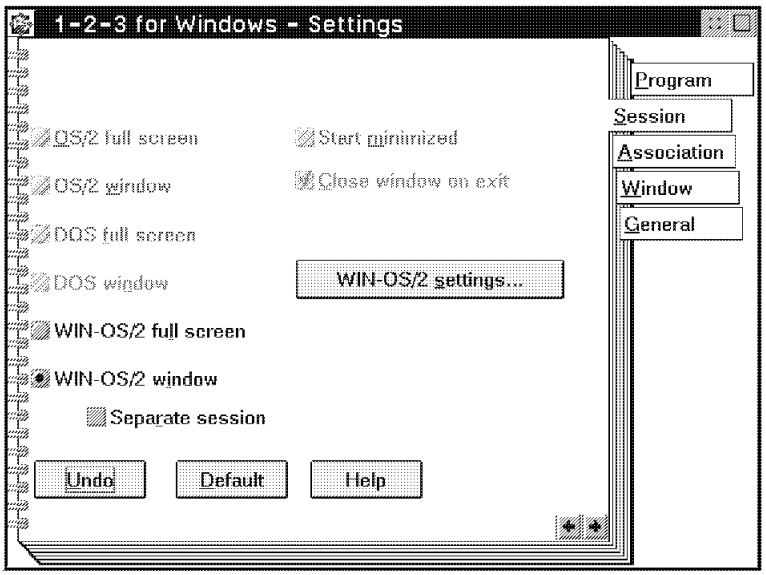


Figure 57. Session Page of a Program Settings Notebook

Table 16. WPProgram Session Setup String Parameters

Keyname	**Value**	**Description**
MINIMIZED	YES	Start program minimized
MAXIMIZED	YES	Start program maximized
NOAUTOCLOSE	YES	Leaves the window open upon program termination.
	NO	Closes the window when the program terminates.
PROGTYPE	value	Sets the session value. See Table 17 on page 70 for the possible values.
SET	X=V	X is any environment variable. V sets the value of the environment variable. When used will wipe out many variables you may assumed were set. Check environment space closely when using. Also used to specify DOS settings for DOS and Windows programs. See the Table 18 on page 72 for DOS and WIN-OS2 settings.

The MINIMIZED and MAXIMIZED settings only have one value, and that is YES. If you don't specify these settings the window will open normally. If you include both these settings in the Value parameter then only the last setting will determine the way the program will open. MINIMIZED=YES will have no effect on PM programs as you cannot check the Start Minimized button in the settings notebook for PM programs. The following example creates an OS/2 Window that will open minimized.

```
"PM_InstallObject" "OS/2 Window;WPProgram;<WP_DESKTOP>"
   "EXENAME=*;PROGTYPE=WINDOWABLEVIO;MINIMIZED=YES;
   OBJECTID=OS2WIN"
```

Figure 58. Setting to Open a Window Minimized

Table 17. WPProgram Session Setup String Parameters for PROGTYPE=	
Value	**Description**
OS/2 session values:	
PM	Sets the session type to PM
FULLSCREEN	Sets the session type to OS/2 full screen
WINDOWABLEVIO	Sets the session type to OS/2 windowed
DOS session values:	
VDM	Sets the session type to DOS full screen
WINDOWEDVDM	Sets the session type to DOS windowed
WIN-OS/2 session values:	
WIN	WIN-OS2 full screen
WINDOWEDWIN	WIN-OS2 windowed, NOT a separate VDM session
SEPARATEWIN	WIN-OS2 windowed, Separate VDM session
OS/2 2.1 systems session values:	
PROG_31_STD	WIN-OS2 full screen, Windows 3.1 Standard mode.
PROG_31_STDSEAMLESSVDM	WIN-OS2 windowed, Separate VDM session, 3.1 Standard mode
PROG_31_STDSEAMLESSCOMMON	WIN-OS2 windowed, NOT a separate VDM session, 3.1 Standard mode
PROG_31_ENH	WIN-OS/2 full screen, NOT a separate VDM session, 3.1 Enhanced Compatibility
PROG_31_ENHSEAMLESSVDM	WIN-OS2 windowed, Separate VDM session, 3.1 Enhanced Compatibility
PROG_31_ENHSEAMLESSCOMMON	WIN-OS2 windowed, NOT a separate VDM session, 3.1 Enhanced Compatibility

The value SET X=V referred to in Table 16 on page 68 is the DOS and Windows settings that are found in the Sessions page in the Program settings. Table 18 on page 72 lists the settings for DOS and Windows settings. They are used in the RC files with a SET keyword, and then the DOS/Window setting keyname with its value. For example:

```
"PM_InstallObject" "Special DOS;WPProgram;<WP_DESKTOP>"
   "EXENAME=*;PROGTYPE=WINDOWEDVDM;
   SET HW_NOSOUND=ON;SET KBD_CTRL_BYPASS=ALT_ESC;
   SET XMS_MEMORY_LIMIT=6144;
   OBJECTID=SPECIALDOS"
```

Important Notes on DOS and WIN-OS2 Settings

- To change these values you use: keyname=value. For example:

  ```
  SET DOS_FILES=45;SET DOS_HIGH=1;
  ```

  Also note that on the settings page you click on ON or OFF for some values. From an .RC file you use 1 for ON and 0 for OFF. For example:

  ```
  SET COM_HOLD=1;
  ```

  sets to ON to keep the communications ports open until the session ends.

- Some settings may already have default values, like DOS_VERSION. You must be careful since any action against that setting is treated as a replacement (even if you are using the updateifexist duplicate flag value). So if you wanted to add one item to DOS_VERSION, you should also include all of the existing values.
- Some settings are new once you've installed the OS/2 V2 Service Pack or upgraded to OS/2 V2.1. As well some may not be on your workstation due to your hardware configuration, for instance use of VIDEO_8514A_XGA_IOTRAP is only available on certain systems.
- WIN-OS2 Settings are new to V2 users and appear once the Service Pack is installed or you have upgraded to OS/2 V2.1.

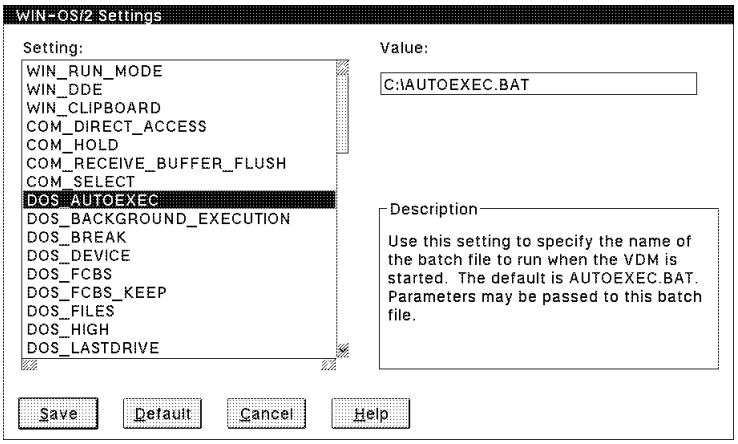


Figure 59. Settings Dialog on the Session Page of a Program Settings Notebook

Table 18 (Page 1 of 5). DOS and WIN-OS2 Settings Fields <default>

Keyname	**Value**
WIN_RUNMODE	Use the PROGTYPE parameter mentioned previously to define a enhanced mode WIN-OS2 program (Also see note 4 from above)
WIN_DDE	(See note 4 above)
WIN_CLIPBOARD	(See note 4 above)
AUDIO_ADAPTER_SHARING	1 0
COM_DIRECT_ACCESS	1 < 0 >
COM_HOLD	1 < 0 >
COM_RECEIVE_BUFFER_FLUSH	Valid settings: <NONE> ALL RECEIVE DATA INTERRUPT ENABLE SWITCH TO FOREGROUND
COM_SELECT	Valid settings: <ALL> COM1 COM2 COM3 COM4 NONE
DOS_AUTOEXEC	C:AUTOEXEC.BAT Use full BATch filename also you can pass parameters
DOS_BACKGROUND_EXECUTION	< 1 > 0

Table 18 (Page 2 of 5). DOS and WIN-OS2 Settings Fields <default>	
Keyname	**Value**
DOS_BREAK	1 < 0 >
DOS_DEVICE	Default: empty Remember to separate any used with "," for new line
DOS_FCBS	Limits: 0-255, default 16
DOS_FCBS_KEEP	Limits: 0-255, default 8
DOS_FILES	Limits: 20-255, default 20
DOS_HIGH	1 < 0 >
DOS_LASTDRIVE	Limits: last physical drive to Z, default Z
DOS_RMSIZE	Limits: 128-640, default 640, increments of 16
DOS_SHELL	Default: ″?:OS2MDOSCOMMAND.COM ″ ″?:OS2MDOS /P″ where ? is the boot drive
DOS_STARTUP_DRIVE	Default: empty. Accepts text like A: or C:DISKSDRDOS.IMG
DOS_UMB	1 < 0 >
DOS_VERSION	Default: DCJSS02.EXE,3,40,255 DFIA0MOD.SYS,3,40,255 DXMA0MOD.SYS,3,40,255 IBMCACHE.COM,3,40,255 IBMCACHE.SYS,3,40,255 ISAM.EXE,3,40,255 ISAM2.EXE,3,40,255 ISQL.EXE,3,40,255 NET3.COM,3,40,255 EXCEL.EXE,10,10,4 PSCPG.COM,3,40,255 SAF.EXE,3,40,255 WIN200.BIN,10,10,4 Remember what you put here will replace the existing list of items so be careful. Also remember to use a caret (Shift 6) in front of any commas you need. Example: `SET DOS_VERSION=IBMCACHE.SYS^,3^,40^,255`

Table 18 (Page 3 of 5). DOS and WIN-OS2 Settings Fields <default>	
Keyname	**Value**
DPMI_DOS_API	Valid settings: <AUTO> ENABLED DISABLED
DPMI_MEMORY_LIMIT	Limits: 0-512, default 4
DPMI_NETWORK_BUFF_SIZE	Limits: 1-64, default 8
EMS_FRAME_LOCATION	Valid settings: <AUTO> NONE C000 C400 C800 CC00 D000 D400 D800 DC00 8000 8400 8800 8C00 9000
EMS_HIGH_OS_MAP_REGION	Limits: 0-96, default 32, note increments of 16
EMS_LOW_OS_MAP_REGION	Limits: 0-576, default 384, note increments of 16
EMS_MEMORY_LIMIT	Limits: 0-32768, default 2048, note increments of 16
HW_NOSOUND	1 < 0 >
HW_ROM_TO_RAM	1 < 0 >
HW_TIMER	1 < 0 >
IDLE_SECONDS	Limits: 0-60, default 0
IDLE_SENSITIVITY	Limits: 1-100, default 75
INT_DURING_IO	1 < 0 >
KBD_ALTHOME_BYPASS	1 < 0 >
KBD_BUFFER_EXTEND	< 1 > 0
KBD_CTRL_BYPASS	Valid settings: <NONE> ALT_ESC CTRL_ESC
KBD_RATE_LOCK	1 < 0 >
MEM_EXCLUDE_REGIONS	Initially empty. You can specify a range of memory to exclude or you can supply a singe address for the beginning of a 4KB region, if you need several regions separate them with a comma (don't forget to use a caret (Shift 6) since commas are special setup string parameters) Example: `SET MEM_EXCLUDE_REGIONS=C0000^,D0000-D8000`

Table 18 (Page 4 of 5). DOS and WIN-OS2 Settings Fields <default>	
Keyname	**Value**
MEM_INCLUDE_REGIONS	Initially empty. You can specify a range of memory to include or you can supply a singe address for the beginning of a 4KB region, if you need several regions separate them with a comma (don't forget to use a caret (Shift 6) since commas are special setup string parameters) Example: `SET MEM_INCLUDE_REGIONS=C0000^,D0000-D7FFF` NOTE: The include region D0000-D8000 will include the entire memory between D8000 and D8FFFF.
MOUSE_EXCLUSIVE_ACCESS	1 < 0 >
NETWARE_RESOURCES	Valid settings: NONE PRIVATE GLOBAL Special note: you use the words to change the value BUT the string MUST be 7 characters long! Example: `SET NETWARE_RESOURCES=GLOBAL`
PRINT_SEPARATE_OUTPUT	< 1 > 0
PRINT_TIMEOUT	Limits: 0-3600, default 15
TOUCH_EXCLUSIVE_ACCESS	1 0
VIDEO_8514A_XGA_IOTRAP	< 1 > 0
VIDEO_FASTPASTE	1 < 0 >
VIDEO_MODE_RESTRICTION	Valid settings: <NONE> CGA MONO Special note, you use the words to change the value BUT the string MUST be 15 characters long! Example: `SET VIDEO_MODE_RESTRICTION=CGA`
VIDEO_ONDEMAND_MEMORY	< 1 > 0
VIDEO_RETRACE_EMULATION	< 1 > 0
VIDEO_ROM_EMULATION	< 1 > 0
VIDEO_SWITCH_NOTIFICATION	1 < 0 >
VIDEO_WINDOW_REFRESH	Limits: 1-600, default 1
XMS_HANDLES	Limits: 0-128, default 32

Table 18 (Page 5 of 5). DOS and WIN-OS2 Settings Fields <default>	
Keyname	**Value**
XMS_MEMORY_LIMIT	Limits: 0-16384, default 2048, increment of 4
XMS_MINIMUM_HMA	Limits: 0-63, default 0

Associations are used to create a relationship between a data file and a program. Usually the program created the data files or the program can load the data files. For example there exists an association between the icon files with the association “Icon” and extension .ICO and the Icon editor.

Some examples of Association Types are Plain Text, Icon, and Executable. If you open the settings for a program or a data file you can see the list of associations on the Associations page of the Settings notebook. The association filter is the actual filenames that the program will be associated with. Wildcards can be used for the association filters (see Figure 60).

```
"PM_InstallObject" "OS/2 System Editor;WPProgram;<WP_TOOLS>;UPDATE"
   "EXENAME=?:\OS2\E.EXE;PROGTYPE=PM;
    ASSOCTYPE=Plain Text,OS/2 Command File,DOS Command File,,;
    ASSOCFILTER=*.DOC,*.TXT,,;
    OBJECTID=<WP_SYSED>"
```

Figure 60. Setting Associations for the Program Object

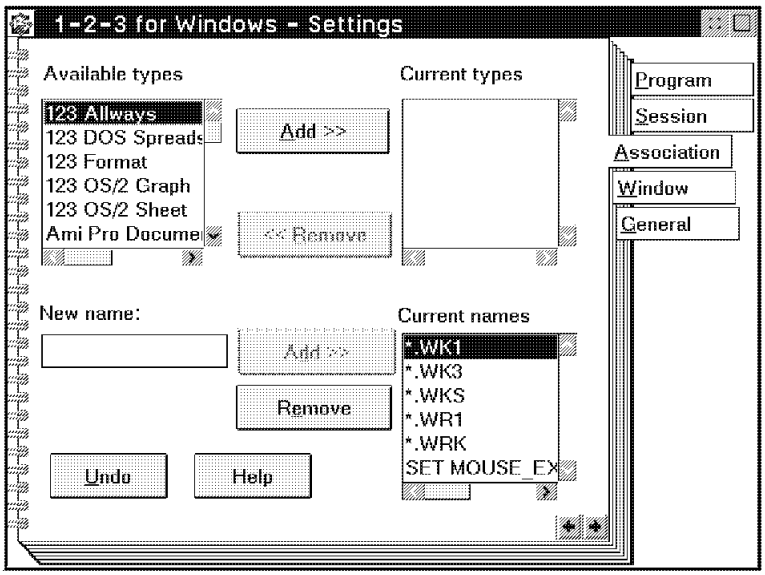


Figure 61. Association Page of a Program Settings Notebook

Table 19. WPProgram Association Setup String Parameters

Keyname	Value	Description
ASSOCFILTER	filters	Sets the filename filter for files associated to this program. Multiple filters are separated by commas. See notes about preserving existing filter values following.
ASSOCTYPE	type	Sets the type of files associated to this program. Multiple filters are separated by commas. See notes about preserving existing associate types following.

Note

When using ASSOCFILTER and/or ASSOCTYPE include two commas at the end of your string to preserve any settings that are already applied. For example:

```
ASSOCTYPE=Metafile,PIF file,,;ASSOCFILTER=*.MET,*.PIF,,;
```

The following settings are the settings in the Window page of a program object settings. If a program is programmed to hide then the setting to minimized the program to the viewer or to the desktop will be ignored. CCVIEW selects if more than one program can be opened from the same object on the desktop. This may only apply to this one object and may change when the global settings are changed.

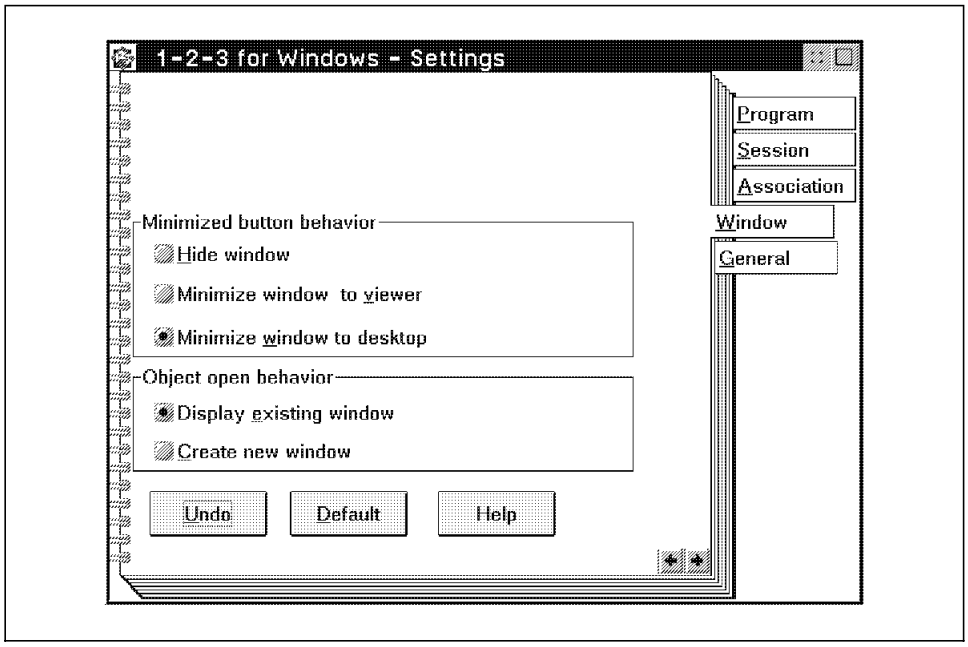


Figure 62. Window Page of a Program Settings Notebook

Table 20. WPProgram Window Setup String Parameters		
Keyname	**Value**	**Description**
MINWIN	HIDE	Views of this object will hide when their minimize button is selected.
	VIEWER	Views of this object will minimize to the minimized window viewer when their minimize button is selected.
	DESKTOP	Views of this object will minimize to the Desktop when their minimize button is selected.
CCVIEW	YES	New views of this object will be created every time the user selects open.
	NO	Open views of this object will resurface when the user selects open.

General and Miscellaneous Table: Don't use quotes or brackets to enclose the filename for the ICONFILE and HELPFILENAME settings.

```
"PM_InstallObject" "Folder;WPFolder;<WP_FOLDER>"
   "ICONFILE=D:\ICONS\CALVIN.ICO;OBJECTID=FOLDER"
```

To use the ICONRESOURCE you need to know the icon's ID in the DLL file. As mentioned before, this information can be obtained using a REXX program. The ICONPOS and ICONVIEWPOS are resolution independent. They work on a percentage value, where is 0 0 is the bottom left corner and 100 100 is the top right corner of any display. This example places the object icon in the middle of the screen, and opens the folder in the bottom right quarter of the screen.

```
"PM_InstallObject" "Command;WPProgram;<WP_DESKTOP>"
   "ICONPOS=50 50;ICONVIEWPOS=50 0 50 50;OBJECTID=COMMAND"
```

If you enclose the TITLE in quotes the quotes will be part of the new title. The last number of settings are all flags and determine the behavior of the object and what is in the object's system menu. Be very cautious with settings such as NOTVISIBLE=YES for obvious reasons.

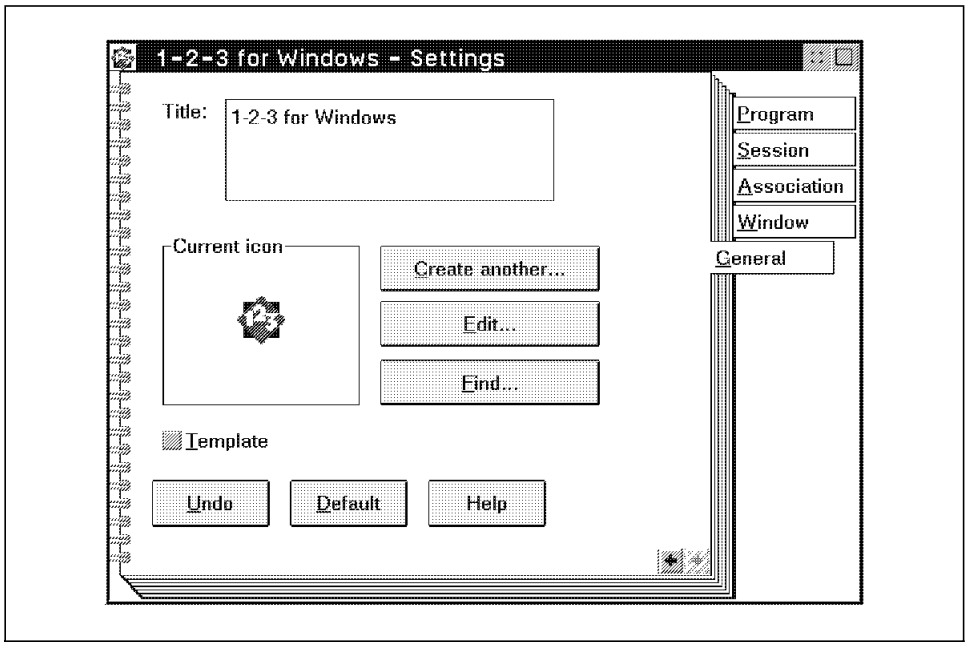


Figure 63. General Page of a Program Settings Notebook

Table 21. WPFolder General Setup String Parameters		
Keyname	**Value**	**Description**
TEMPLATE	YES	Creates object as a template.
	NO	Resets objects template property.
TITLE	value	Can be used to assign a name/title to an object.

Table 22. WPFolder Icon Related Setup String Parameters

Keyname	Value	Description
ICONFILE	filename	This sets the object's icon.
ICONRESOURCE	id module	This sets the object's icon. 'id' is the identity of an icon resource in the 'module' dynamic link library (DLL). For example: `ICONRESOURCE=73 PMWP;` This would indicate resource 73 in PMWP.DLL.
ICONPOS	l% b%	This sets the object's initial icon position. The l and b values represent the position in the object's folder in percentage coordinates.
ICONVIEWPOS	l% b% w% h%	This sets the object's initial size. The values represent relative position in percentage coordinates. For example: ICONPOS=25 25 50 50 would create a folder whose bottom left corner is 25% from the left and 25% from the bottom and half the screen width/height.

Table 23. WPFolder Miscellaneous Setup String Parameters

Keyname	Value	Description
OBJECTID	<name>	This sets the object's identity. The object ID will stay with the object even if it is moved or renamed. An object ID is any unique string preceded with a '<' and terminated with a '>'. This may also be a real name specified as a fully qualified path name. IMPORTANT: For any object you create you should use a unique object ID! Do this for two reasons: 1. If you use an object ID it signifies a unique object that will not be recreated if you use the "FailIfExists" flag in your REXX call. Not using an object ID would cause multiple objects to be created if the same program were run over and over. 2. Should you need to later delete it or change your object, you can use this object ID in your REXX call to refer to it. Also one should not use an object ID that starts with "WP_" as many OS/2 objects use those, consider those reserved characters.
HELPPANEL	id	This sets the object's default help panel.
HELPLIBRARY	filename	This sets the help library.
OPEN	SETTINGS	Open settings view of object when created/updated.
	DEFAULT	Open default view of object when created/updated. Don't forget for folder objects you can use OPEN with these values: ICON, TREE, DETAILS

Table 24. WPFolder Object Properties Setup String Parameters

Keyname	Value	Description
NODELETE	YES	Will not allow you to delete the object.
	NO	Resets the object's no delete property.
NOCOPY	YES	Will not allow you to make a copy.
	NO	Resets the object's no copy property.
NOMOVE	YES	Will not allow you to move the object to another folder, will create shadow on a move.
	NO	Resets the object's no move property.
NODRAG	YES	Will not allow you to drag the object.
	NO	Resets the object's no drag property.
NOLINK	YES	Will not allow you to create a shadow link.
	NO	Resets the object's no link property.
NOSHADOW	YES	Will not allow you to create a shadow link.
	NO	Resets the object's no shadow property.
NORENAME	YES	Will not allow you to rename the object.
	NO	Resets the object's no rename property.
NOPRINT	YES	Will not allow you to print it.
	NO	Resets the object's no print property.
NOTVISIBLE	YES	Will not display the object.
	NO	Resets the object's not visible property.

Creating Shadow Objects: A shadow object is a persistent link or reference to another object. This is achieved by storing away the location and identity of the object that it is linked to and then rerouting all requests for help, context menus, and open views on to the object that is linked to. Delete, Copy, and Move are the only action requests that are handled by the WPShadow object and are not rerouted to the linked object.

When you place objects into your Startup folder you should be placing a shadow object there. See the SHADOW sample code for an example of its usage.

Note

When creating a shadow object always use SHADOWID= to give your object a unique ID so that later you may refer to it, and potentially delete it with SysDestroyObject (See Figure 64).

Table 25. WPShadow Setup String

Keyname	Value	Description
SHADOWID	<name> or filename	The value for this is an object's ID (OBJECTID) or a fully qualified pathname of a directory, program file, or data file.

```
"PM_InstallObject" "Enhanced Editor;WPShadow;<WP_DESKTOP>"
"SHADOWID=<WP_EPM>"
```

Figure 64. Example of a Shadow Object

3.5.6 Color and Display Settings

The last part of the user .RC file is the color settings. The first line informs PM_Colors the type of display you are using. Thereafter PM_Colors and PM_Default_Colors alternate with the same Key value. The PM_Colors setting will be the actual color that will be displayed on the desktop after startup. The PM_Default_Colors setting is the color that the object will change to when you press the Default button. The Default button referred to is the button found in the Edit Scheme dialog box when you edit a scheme in the Scheme Palette. The values are in the RGB(Red-Green-Blue) format. The first value is the red value, the second the green value and the third the blue value. The values are in the range of 0 to 255, where 0 is the lowest

intensity and 255 is the highest intensity. A value of “0 0 0” is black, “255 0 0” is red, “255 255 255” is white and so on. The values must be separated by at least one space.

On a resolution that supports only 16 colors you will have 16 solid colors and all the other color values are dithers of those 16 colors. On resolutions that support 256 colors or more each value will give a solid color. For more information on determining color numbers see 4.3.3.1, “Workplace Shell Color Values” on page 166.

```
"PM_Colors"          "Background"           "  0   0   0"
"PM_Colors"          "IconText"             "255 255 255"
```

Figure 65. Set the Background to Black and IconText to White

The last three entries are the device drivers for the display adapter card. Unless you know the drivers for other display adapters, we suggest that you don’t change the drivers. It is the drivers that make the portability of the .RC files difficult. One way of using .RC files created on another system that has a different display is by editing the INI.RC file on the target machine and changing the first and last three lines in this section to look the same as the target machine. In the following example we list the first line and the last three lines of this section to show the difference between a VGA and an XGA system.

```
VGA System
"PM_Colors"          "Display"              "VGA"

"PM_DISPLAYDRIVERS"  "IBMXGA32"             "IBMVGA32"
"PM_DISPLAYDRIVERS"  "CURRENTDRIVER"        "IBMVGA32"
"PM_DISPLAYDRIVERS"  "DEFAULTDRIVER"        "IBMVGA32"

XGA System
"PM_Colors"          "Display"              "XGA"

"PM_DISPLAYDRIVERS"  "IBMXGA32"             "IBMXGA32"
"PM_DISPLAYDRIVERS"  "CURRENTDRIVER"        "IBMXGA32"
"PM_DISPLAYDRIVERS"  "DEFAULTDRIVER"        "IBMXGA32"
```

Figure 66. Difference Between VGA and XGA Systems

3.6 The System.RC File Structure

There is not much you can change in the System.RC file. It contains some information about the printers, spoolers, ports, and the color settings for the Scheme Palette.

```
"PM_INFO"           "Version"          "2.1"
"PM_SPOOLER"        "SPOOL"            "1;"
"PM_SPOOLER"        "DIR"              "C:\SPOOL;"
"PM_SPOOLER_QP"     "PMPRINT"          "C:\OS2\DLL\PMPRINT.QPR;;"
"PM_SPOOLER_DD"     "IBMNULL"          "IBMNULL.DRV;;;"
"PM_SPOOLER_PORT"   "LPT1"             ";"
"PM_SPOOLER_PORT"   "LPT2"             ";"
"PM_SPOOLER_PORT"   "LPT3"             ";"
"PM_SPOOLER_PORT"   "COM1"             "9600;0;8;1;1;"
"PM_SPOOLER_PORT"   "COM2"             "9600;0;8;1;1;"
"PM_SPOOLER_PORT"   "COM3"             "9600;0;8;1;1;"
"PM_SPOOLER_PORT"   "COM4"             "9600;0;8;1;1;"
```

Figure 67. The System Information of INISYS.RC

The System.RC file is called INISYS.RC. As with the User.RC file, the System.RC file starts with the RC file header. It has the same line as the User.RC file for the version number.

```
"PM_INFO" "Version" "2.1"
```

The spooler information is next. The Key value SPOOL allows you to specify the number of spoolers for your system. The directory of the spooler can be changed by using a different pathname for the DIR Key value. PM_SPOOLER_QP is the queue printer driver for OS/2 and we suggest that this is not changed.

When you add printer drivers in the user .INI file you need to update the spooler to recognize the new printer driver. This is done with the application PM_SPOOLER_DD. You just add the printer driver name for the Key parameter and the printer driver filename for the Value parameter. You can add more printer ports and communication ports to the system. For example, if you want to add three additional LPT ports for LAN or Redirection use then add the following lines:

```
"PM_SPOOLER_PORT"   "LPT4"             ";"
"PM_SPOOLER_PORT"   "LPT5"             ";"
"PM_SPOOLER_PORT"   "LPT6"             ";"
```

You can also change the settings for a COM port. The first value in the data value is the baud rate, the second the parity, then the word length, the fourth the stop bits and the last value is the handshake. In the example below we change COM3 port to 2400 baud and changed COM4 to 1200 baud, no parity, 7 word length, 1 stopbit and hardware handshake.

```
"PM_SPOOLER_PORT"       "COM3"              "2400;0;8;1;1;"
"PM_SPOOLER_PORT"       "COM4"              "1200;1;7;1;0;"
```

3.7 Examples of RC Files

We are going to include some examples where all the different features of the .RC files are used together. We are going to use the example of a user and an administrator desktop, similar to the example at the begining of this book, to demonstrate how to change the desktops using the .RC files.

We tested the .RC file on both XGA and VGA displays. If you are going to use these examples please change the lines containing the settings for the display driver to the display driver that you are using. See 3.5.6, “Color and Display Settings” on page 84 for more information on selecting the correct display drivers.

These icon positions are set for an XGA or VGA display. Because we are working with relative positions and not fixed positions the icons may not be ideally positioned for an EGA display or displays that use a resolution different to that of XGA or VGA. The resolution on an XGA is 1024x768 pixels and on a VGA 640x480 pixels. What we have found is that the icon text tends to run over the edges of the desktop, resulting in scroll bars on the desktop, and overlapping icons. The reason for this is there are more pixels between icons on an XGA than on a VGA because of the higher resolution on an XGA.

3.7.1 Creating a New Desktop

The first set of examples will be where we use an .RC file to create a new desktop. These files need to be compiled to a new .INI file for the correct results. If you want to compile the file to an file name that exists you have to delete the old file. If you don’t delete the old file you will get the objects that were on the old desktop as well as the objects that were programmed in the new .RC file. See 3.3, “The MAKEINI Compiler” on page 21 for more information on compiling the .RC files.

3.7.1.1 Creating a Simple Desktop

In the following example we have created an .RC file that has a very limited use. This desktop is designed for users who will only use a spreadsheet, a word processor and a graphics application. For each of the applications we created associations so that we can use the data files to open the applications, instead of loading the applications and then opening the data files. These associations are:

- **Word processor** for Describe
- **Spreadsheet** for Lotus** 1-2-3* *
- **Drawings** for Freelance**

We created folders which will contain all the data files, named Word processing, Spreadsheets and Drawings. In each folder will be a new file with the correct extensions, and associations for each application. All the icons in the folders will be in a flowed view format. We created a fourth folder called layouts, which will have the templates for the relevant data files.

We create a program object for each of these applications. Because the user doesn't need to run these applications from the program object we make the object not visible with the NOTVISIBLE=YES flag. The applications we selected are DeScribe for word processing, Lotus 1-2-3 for Windows for spreadsheet and Freelance Graphics for drawing.

We removed all the other objects and folders from the desktop as the users will have no need for them. We set the data files and templates so that they could not be deleted, moved, copied, or shadowed. We set all the open windows to be minimized to the desktop. Following is a list of the changes we have made as they appear in the listing of SMALDESK.RC. The changed items described in the text are all in bold letters in the listings of SMALDESK.RC.

Note

We decided to use Lotus 1-2-3 for Windows to show additional flexibility in using OS/2, Windows and DOS programs. There is an OS/2 version of Lotus 1-2-3 available.

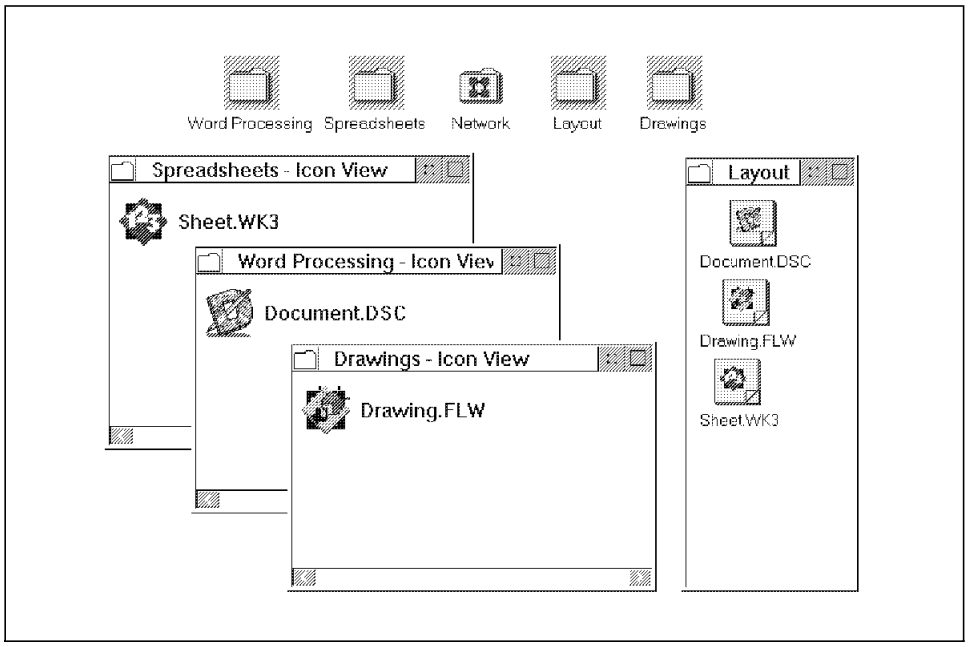


Figure 68. A Simple Desktop Created With SMALDESK.RC

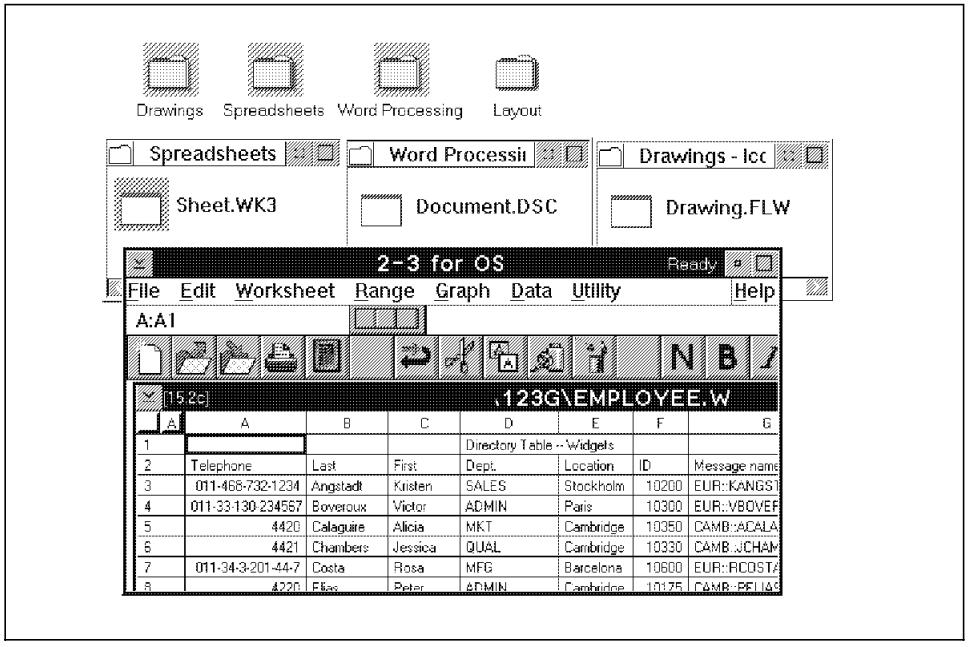


Figure 69. Small Desktop with SHEET.WK3 Open

```
/* File used by MAKEINI.EXE to create a user.INI                      */
/* Use the command                                                    */
/*    MAKEINI NEW.INI SMALDESK.RC                                     */
/* to create the file NEW.INI.                                        */
/* To copy over the OS2.INI, use a DOS boot diskette or the           */
/* OS2 Install diskette ESCed to the command line to copy NEW.INI     */
/* over the existing OS2.INI that is to be replaced.                  */
/* To use NEW.INI change the SET USER_INI=OS2.INI line in CONFIG.SYS  */
/* to SET USER_INI=NEW.INI.                                           */
/* Note that if NEW.INI already exists, then it is updated.           */

CODEPAGE 850

STRINGTABLE
BEGIN

  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""

  "PM_ControlPanel"    "BorderWidth"          "4"
  "PM_ASSOC_TYPE"      "Spreadsheet"          ""
  "PM_ASSOC_TYPE"      "Wordprocessor"        ""
  "PM_ASSOC_TYPE"      "Drawings"             ""
```

Figure 70. SMALDESK.RC File Header and Associations (Part 1 of 7)

```
"PM_DEVICE_DRIVERS"  "IBMNULL"      "C:\OS2\DLL\IBMNULL\IBMNULL.DRV"
"PM_Font_Drivers"    "PMATM"        "\OS2\DLL\PMATM.DLL"
"PM_Fonts"           "SYSMONO"      "\OS2\DLL\SYSMONO.FON"
"PM_Fonts"           "COURIER"      "\OS2\DLL\COURIER.FON"
"PM_Fonts"           "HELV"         "\OS2\DLL\HELV.FON"
"PM_Fonts"           "TIMES"        "\OS2\DLL\TIMES.FON"
"PM_Fonts"           "COURIERI"     "\OS2\DLL\COURIERI.FON"
"PM_Fonts"           "HELVI"        "\OS2\DLL\HELVI.FON"
"PM_Fonts"           "TIMESI"       "\OS2\DLL\TIMESI.FON"
"PM_Fonts"           "MARKSYM.OFM"  "\PSFONTS\MARKSYM.OFM"
"PM_Fonts"           "HELV.OFM"     "\PSFONTS\HELV.OFM"
"PM_Fonts"           "HELVB.OFM"    "\PSFONTS\HELVB.OFM"
"PM_Fonts"           "HELVBI.OFM"   "\PSFONTS\HELVBI.OFM"
"PM_Fonts"           "HELVI.OFM"    "\PSFONTS\HELVI.OFM"
"PM_Fonts"           "COUR.OFM"     "\PSFONTS\COUR.OFM"
"PM_Fonts"           "COURB.OFM"    "\PSFONTS\COURB.OFM"
"PM_Fonts"           "COURBI.OFM"   "\PSFONTS\COURBI.OFM"
"PM_Fonts"           "COURI.OFM"    "\PSFONTS\COURI.OFM"
"PM_Fonts"           "SYMB.OFM"     "\PSFONTS\SYMB.OFM"
"PM_Fonts"           "TNR.OFM"      "\PSFONTS\TNR.OFM"
"PM_Fonts"           "TNRB.OFM"     "\PSFONTS\TNRB.OFM"
"PM_Fonts"           "TNRBI.OFM"    "\PSFONTS\TNRBI.OFM"
"PM_Fonts"           "TNRI.OFM"     "\PSFONTS\TNRI.OFM"

"PM_INFO"            "Version"            "2.1"
"PM_IMAGECNV"        "IMAGECNVPATH"       "\OS2\IMAGECNV"
"SYS_DLLS"           "LoadOneTime"        "REXXINIT"
"SYS_DLLS"           "LoadPerProcess"     "PMCTLS"
"PM_SPOOLER"         "QUEUE"              "LPT1Q;"
"PM_SPOOLER"         "PRINTER"            "PRINTER1;"
```

Figure 71. SMALDESK.RC Fonts and System Settings (Part 2 of 7)

```
  /* pm_national temporarily back in */
  "PM_National"  "iCountry"   "1" /* Country code (phone ID of country)  */
  "PM_National"  "iDate"      "0" /* Date mode (0:MDY, 1:DMY, 2:YMD)     */
  "PM_National"  "iCurrency"  "0" /* Currency mode 0: prefix, no sep     */
                                  /*    1: suffix, no separation         */
                                  /*    2: prefix, 1 CHAR separation     */
                                  /*    3: suffix, 1 CHAR separation     */
  "PM_National"  "iDigits"       "2" /* Signif Decimal digits in Currency*/
  "PM_National"  "iTime"         "0" /* time mode (0=12 hours clock,1=24)*/
  "PM_National"  "iLzero"        "0" /* Leading zeros (0: no, 1: yes)    */
  "PM_National"  "s1159"         "AM"/* Trailing string 0:00 to 11:59    */
  "PM_National"  "s2359"         "PM"/* Trailing string 12:00 to 23:59   */
  "PM_National"  "sCurrency"     "$" /* Currency Symbol string           */
  "PM_National"  "sThousand"     "," /* Thousands separator string       */
  "PM_National"  "sDecimal"      "." /* Decimal separator string         */
  "PM_National"  "sDate"         "-" /* Date separator string            */
  "PM_National"  "sTime"         ":" /* time separator string            */
  "PM_National"  "sList"         "," /* List separator string.           */
  "PM_National"  "iMeasurement"  "1" /* 1=English, 2=Metric,             */
                                     /* 3=Points, 4=Pica                 */

  "PM_IBMBGA"    "ALTSYSFONT" "0"  /* For 8514 adapter only */
  "PM_IBMBGA"    "FASTSS"     "0"  /* For 8514 adapter only */

  "EPM"          "EPMIniPath" "\OS2\EPM.INI"
  "PMDiary"      "IniPath"    "\OS2\PMDIARY.INI"
/* Change default for Win-OS/2 Setup Object - R206 60555 */
  "WINOS2"       "PM_GlobalWindows31Settings"
                 "DPMI_MEMORY_LIMIT=64;PROGTYPE=PROG_31_STD;
                 KBD_ALTHOME_BYPASS=1;VIDEO_SWITCH_NOTIFICATION=1;
                 VIDEO_8514A_XGA_IOTRAP=0"

  "PM_Workplace:InstallGroups" "1" "1"
  "PM_DefaultSetup" "MINWIN" "DESKTOP"

  "PM_InstallObject" "Small Desktop;WPDesktop;?:\" "OBJECTID=<WP_DESKTOP>"

  /* remove PMDDE object from workplace */
  "PM_InstallObject" "Deleted;WPProgram;<WP_TOOLS>;DELETE"
                     "OBJECTID=<WP_PMDDE>"
END
```

Figure 72. SMALDESK.RC National and System Settings (Part 3 of 7)

```
STRINGTABLE REPLACEMODE
BEGIN
  "PM_InstallObject" "Nowhere;WPFolder;?:\"  "OBJECTID=<WP_NOWHERE>"
  "PM_InstallObject" "Printer;PDView;<WP_DESKTOP>"
                     "Printer;OBJECTID=<WP_PDVIEW>"
  "PM_InstallObject" "Network;WPNetwork;<WP_DESKTOP>"
                     "HELPPANEL=30000;NODELETE=YES;OBJECTID=<WP_NETWORK>"
  "PM_InstallObject" "DeScribe;WPProgram;<WP_DESKTOP>"
                     "EXENAME=?:\DESCRIBE\DESCRIBE.EXE;PROGTYPE=PM;
                     NOTVISIBLE=YES;ASSOCTYPE=Wordprocessor,,;
                     ASSOCFILTER=*.DSC,,;OBJECTID=<WP_DESCRIBE>"
  "PM_InstallObject" "Lotus 1-2-3;WPProgram;<WP_DESKTOP>"
                     "EXENAME=?:\123W\123W.EXE;PROGTYPE=WINDOWEDWIN;
                     NOTVISIBLE=YES;ASSOCTYPE=Spreadsheet,,;
                     ASSOCFILTER=*.WK?,,;OBJECTID=<WP_LOTUS>"
  "PM_InstallObject" "FreeLance;WPProgram;<WP_DESKTOP>"
                     "EXENAME=?:\FL\FL.EXE;PROGTYPE=WINDOWEDWIN;
                     NOTVISIBLE=YES;ASSOCTYPE=Drawing,,;
                     ASSOCFILTER=*.FLW,,;OBJECTID=<WP_FL>"
  "PM_InstallObject" "Layout;WPFolder;<WP_DESKTOP>"
                     "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                     NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                     ICONVIEWPOS=20 50 60 20;OBJECTID=<WP_LAYOUT>"
    "PM_InstallObject" "Document.DSC;WPDataFile;<WP_LAYOUT>"
                       "TEMPLATE=YES;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                       NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Wordprocessor,,;
                       OBJECTID=<WP_TEMPWORD>"
    "PM_InstallObject" "Sheet.WK3;WPDataFile;<WP_LAYOUT>"
                       "TEMPLATE=YES;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                       NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Spreadsheet,,;
                       OBJECTID=<WP_TEMPSHEET>"
    "PM_InstallObject" "Drawing.FLW;WPDataFile;<WP_LAYOUT>"
                       "TEMPLATE=YES;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                       NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Drawing,,;
                       OBJECTID=<WP_TEMPDRAW>"
```

Figure 73. SMALDESK.RC Installation of Desktop Objects (Part 4 of 7)

```
  "PM_InstallObject" "Word Processing;WPFolder;<WP_DESKTOP>"
                     "ICONVIEW=FLOWED;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                     NODRAG=YES;NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                     ICONVIEWPOS=33 30 30 50;OBJECTID=<WP_WORDFOLDER>"
    "PM_InstallObject" "Document.DSC;WPDataFile;<WP_WORDFOLDER>"
                       "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                       NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Wordprocessor,,;
                       OBJECTID=<WP_WORDFILE>"
  "PM_InstallObject" "Spreadsheets;WPFolder;<WP_DESKTOP>"
                     "ICONVIEW=FLOWED;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                     NODRAG=YES;NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                     ICONVIEWPOS=1 30 30 50;OBJECTID=<WP_SHEETFOLDER>"
    "PM_InstallObject" "Sheet.WK3;WPDataFile;<WP_SHEETFOLDER>"
                       "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                       NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Spreadsheet,,;
                       OBJECTID=<WP_SHEETFILE>"
  "PM_InstallObject" "Drawings;WPFolder;<WP_DESKTOP>"
                     "ICONVIEW=FLOWED;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                     NODRAG=YES;NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                     ICONVIEWPOS=66 30 30 50;OBJECTID=<WP_DRAWFOLDER>"
    "PM_InstallObject" "Drawing.FLW;WPDataFile;<WP_DRAWFOLDER>"
                       "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                       NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Drawing,,;
                       OBJECTID=<WP_DRAWFILE>"
END
```

Figure 74. SMALDESK.RC Installation of Folder Objects (Part 5 of 7)

```
CODEPAGE 850
STRINGTABLE
BEGIN

  "PM_Colors"          "Display"               "XGA"
  "PM_Colors"          "ActiveBorder"          "255 255 128"
  "PM_Colors"          "ActiveTitle"           " 64 128 128"
  "PM_Colors"          "ActiveTitleText"       "255 255 255"
  "PM_Colors"          "ActiveTitleTextBgnd"   " 64 128 128"
  "PM_Colors"          "AppWorkspace"          "255 255 224"
  "PM_Colors"          "Background"            "204 204 204"
  "PM_Colors"          "ButtonDark"            "128 128 128"
  "PM_Colors"          "ButtonDefault"         "  0   0   0"
  "PM_Colors"          "ButtonLight"           "255 255 255"
  "PM_Colors"          "ButtonMiddle"          "204 204 204"
  "PM_Colors"          "DialogBackground"      "204 204 204"
  "PM_Colors"          "EntryField"            "255 255 204"
  "PM_Colors"          "FieldBackground"       "204 204 204"
  "PM_Colors"          "HelpBackground"        "255 255 255"
  "PM_Colors"          "HelpHilite"            "  0 128 128"
  "PM_Colors"          "HelpText"              "  0   0 128"
  "PM_Colors"          "HiliteBackground"      " 96  96  96"
  "PM_Colors"          "HiliteForeground"      "255 255 255"
  "PM_Colors"          "IconText"              "  0   0   0"
  "PM_Colors"          "InactiveBorder"        "204 204 204"
  "PM_Colors"          "InactiveTitle"         "204 204 204"
  "PM_Colors"          "InactiveTitleText"     "128 128 128"
  "PM_Colors"          "InactiveTitleTextBgnd" "204 204 204"
  "PM_Colors"          "Menu"                  "204 204 204"
  "PM_Colors"          "MenuText"              "  0   0   0"
  "PM_Colors"          "MenuHilite"            "204 204 204"
  "PM_Colors"          "MenuHiliteText"        "  0   0   0"
  "PM_Colors"          "MenuDisabledText"      "128 128 128"
  "PM_Colors"          "OutputText"            "  0   0   0"
  "PM_Colors"          "PageBackground"        "255 255 255"
  "PM_Colors"          "Scrollbar"             "192 192 192"
  "PM_Colors"          "Shadow"                "128 128 128"
  "PM_Colors"          "ShadowHiliteBgnd"      "128 128 128"
  "PM_Colors"          "ShadowHiliteFgnd"      "255 255 255"
  "PM_Colors"          "ShadowText"            "128 128 128"
  "PM_Colors"          "TitleBottom"           "128 128 128"
  "PM_Colors"          "TitleText"             "255 255 255"
```

Figure 75. SMALDESK.RC Color and Display Settings (Part 6 of 7)

```
  "PM_Colors"          "Window"                "255 255 255"
  "PM_Colors"          "WindowFrame"           "128 128 128"
  "PM_Colors"          "WindowStaticText"      "  0   0 128"
  "PM_Colors"          "WindowText"            "  0   0   0"
  "PM_DISPLAYDRIVERS"  "IBMXGA32"              "IBMXGA32"
  "PM_DISPLAYDRIVERS"  "CURRENTDRIVER"         "IBMXGA32"
  "PM_DISPLAYDRIVERS"  "DEFAULTDRIVER"         "IBMXGA32"
  ""                   ""                      ""
  "PM_DISPLAYDRIVERS"      "RESOLUTION_CHANGED"      "1"
END
```

Figure 76. SMALDESK.RC Color and Display Settings 2 (Part 7 of 7)

3.7.1.2 Creating a Complex Desktop

In the previous example we created a whole new .RC file to create a new desktop for the system. In the next example we will use the existing INI.RC file, and change the file to be able to create a new desktop. Because most of the windows open in the center of the screen, we move the icons closer to the edge of the desktop. We cannot move them too close to the edge as the Workplace Shell will then automatically place scroll bars at the edge of the desktop. We have added some objects to the desktop and to folders on the desktop, and created shadows on the desktop.

We copied the INI.RC file to another file with a different name. We edited the new file and made the following changes:

- We removed the line to start the Tutorial automatically the first time the desktop is loaded.
- We changed the drive object on the desktop from Drive A to Drive C.
- We created three new program objects on the desktop called:
  - OS/2 Window:1
  - OS/2 Window:2
  - DOS Window
- We set some of the folders, such as OS/2 Windows and the Information folder, to undeletable with NODELETE=YES.
- We moved the Start Here and Master Help Index object from the desktop to the Information folder.
- We created additional setup icons in the System Setup folder called:
  - Display Drivers Install

  - Adobe Type Manager
  - Windows Control Panel
- We created two shadows on the desktop called:
  - Enhanced Editor
  - Drives

The icons' positions are for a display with a resolution of 1024x768. On a display with a different resolution the icons will be in the same place but the text might overlap other icon's text or run off the screen. When the icons' text runs off the edge of the display the Workplace Shell will automatically create scroll bars on the desktop. We tested it on a VGA resolution and we had to change the icon's position by one percent towards the center of the screen to fix the problem. On the left or bottom of the screen you will add and on the top and right you will subtract a percent. For example, the Shredder's position on the XGA is "93 4". To change it for a VGA you change the position to "92 5".

The following figures contain the listing of the file to change the desktop using a resource file called CHNGDESK.RC. The changed items described in the text are all in bold letters in the listings of CHNGDESK.RC.

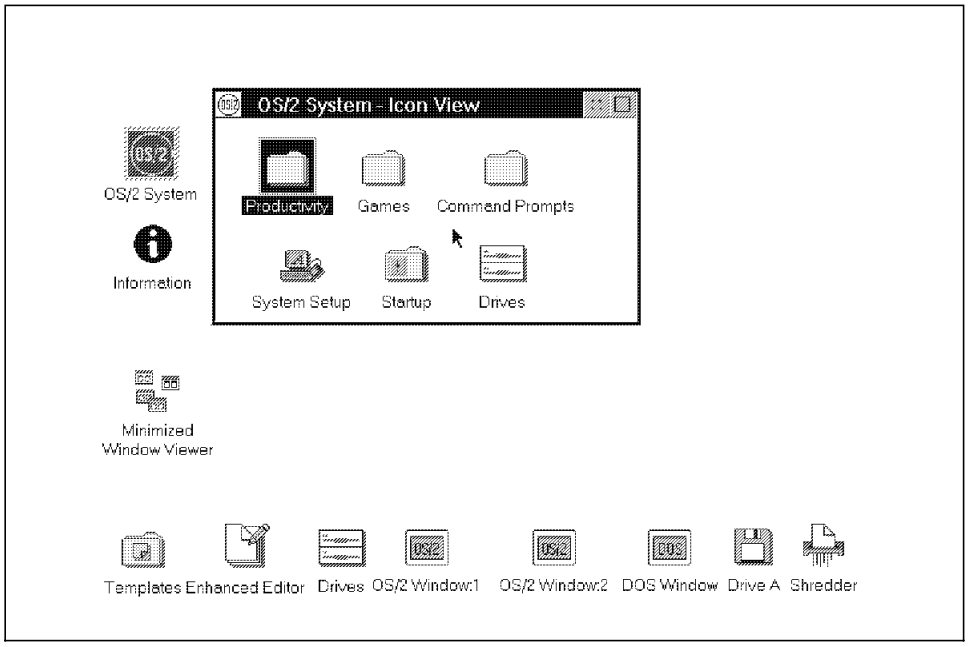


Figure 77. A Complex Desktop Created With CHNGDESK.RC

```
/* File used by MAKEINI.EXE to create user.INI                          */
/* Use the command                                                      */
/*    MAKEINI NEW.INI CHNGDESK.RC                                       */
/* to create the file NEW.INI.                                          */
/* To copy over the OS2.INI, use a DOS boot diskette or the             */
/* OS2 Install diskette ESCed to the command line to copy NEW.INI       */
/* over the existing OS2.INI that is to be replaced.                    */
/* To use NEW.INI change the SET USER_INI=OS2.INI line in CONFIG.SYS    */
/* to SET USER_INI=NEW.INI.                                             */
/* Note that if NEW.INI already exists, then it is updated.             */
CODEPAGE 850
STRINGTABLE
BEGIN

  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  "PM_ControlPanel"    "BorderWidth"          "4"
  "PM_ASSOC_TYPE"      "Plain Text"           ""
  "PM_ASSOC_TYPE"      "OS/2 Command File"    ""
  "PM_ASSOC_TYPE"      "DOS Command File"     ""
  "PM_ASSOC_TYPE"      "Executable"           ""
  "PM_ASSOC_TYPE"      "Metafile"             ""
  "PM_ASSOC_TYPE"      "PIF file"             ""
  "PM_ASSOC_TYPE"      "Bitmap"               ""
  "PM_ASSOC_TYPE"      "Icon"                 ""
  "PM_ASSOC_TYPE"      "Binary Data"          ""
  "PM_ASSOC_TYPE"      "Dynamic Link Library" ""
  "PM_ASSOC_TYPE"      "C Code"               ""
  "PM_ASSOC_TYPE"      "Pascal Code"          ""
  "PM_ASSOC_TYPE"      "BASIC Code"           ""
  "PM_ASSOC_TYPE"      "COBOL Code"           ""
  "PM_ASSOC_TYPE"      "FORTRAN Code"         ""
  "PM_ASSOC_TYPE"      "Assembler Code"       ""
  "PM_ASSOC_TYPE"      "Resource File"        ""
  "PM_ASSOC_TYPE"      "Printer-specific"     ""
```

Figure 78. CHNGDESK.RC File Header and Associations (Part 1 of 12)

```
"PM_DEVICE_DRIVERS"  "IBMNULL"         "C:\OS2\DLL\IBMNULL\IBMNULL.DRV"
"PM_Font_Drivers"    "PMATM"           "\OS2\DLL\PMATM.DLL"
"PM_Fonts"           "SYSMONO"         "\OS2\DLL\SYSMONO.FON"
"PM_Fonts"           "COURIER"         "\OS2\DLL\COURIER.FON"
"PM_Fonts"           "HELV"            "\OS2\DLL\HELV.FON"
"PM_Fonts"           "TIMES"           "\OS2\DLL\TIMES.FON"
"PM_Fonts"           "COURIERI"        "\OS2\DLL\COURIERI.FON"
"PM_Fonts"           "HELVI"           "\OS2\DLL\HELVI.FON"
"PM_Fonts"           "TIMESI"          "\OS2\DLL\TIMESI.FON"
"PM_Fonts"           "MARKSYM.OFM"     "\PSFONTS\MARKSYM.OFM"
"PM_Fonts"           "HELV.OFM"        "\PSFONTS\HELV.OFM"
"PM_Fonts"           "HELVB.OFM"       "\PSFONTS\HELVB.OFM"
"PM_Fonts"           "HELVBI.OFM"      "\PSFONTS\HELVBI.OFM"
"PM_Fonts"           "HELVI.OFM"       "\PSFONTS\HELVI.OFM"
"PM_Fonts"           "COUR.OFM"        "\PSFONTS\COUR.OFM"
"PM_Fonts"           "COURB.OFM"       "\PSFONTS\COURB.OFM"
"PM_Fonts"           "COURBI.OFM"      "\PSFONTS\COURBI.OFM"
"PM_Fonts"           "COURI.OFM"       "\PSFONTS\COURI.OFM"
"PM_Fonts"           "SYMB.OFM"        "\PSFONTS\SYMB.OFM"
"PM_Fonts"           "TNR.OFM"         "\PSFONTS\TNR.OFM"
"PM_Fonts"           "TNRB.OFM"        "\PSFONTS\TNRB.OFM"
"PM_Fonts"           "TNRBI.OFM"       "\PSFONTS\TNRBI.OFM"
"PM_Fonts"           "TNRI.OFM"        "\PSFONTS\TNRI.OFM"

"PM_INFO"            "Version"         "2.1"
"PM_IMAGECNV"        "IMAGECNVPATH"    "\OS2\IMAGECNV"
"SYS_DLLS"           "LoadOneTime"     "REXXINIT"
"SYS_DLLS"           "LoadPerProcess"  "PMCTLS"
"PM_SPOOLER"         "QUEUE"           "LPT1Q;"
"PM_SPOOLER"         "PRINTER"         "PRINTER1;"
```

Figure 79. CHNGDESK.RC Fonts and System Settings (Part 2 of 12)

```
  /* pm_national temporarily back in */
  "PM_National"  "iCountry"   "1" /* Country code (phone ID of country)  */
  "PM_National"  "iDate"      "0" /* Date mode (0:MDY, 1:DMY, 2:YMD)     */
  "PM_National"  "iCurrency"  "0" /* Currency mode 0: prefix, no sep     */
                                  /*    1: suffix, no separation         */
                                  /*    2: prefix, 1 CHAR separation     */
                                  /*    3: suffix, 1 CHAR separation     */
  "PM_National"  "iDigits"       "2"  /* Signif Decimal digits in Currency*/
  "PM_National"  "iTime"         "0"  /* time mode (0=12 hours clock,1=24)*/
  "PM_National"  "iLzero"        "0"  /* Leading zeros (0: no, 1: yes)   */
  "PM_National"  "s1159"         "AM" /* Trailing string 0:00 to 11:59   */
  "PM_National"  "s2359"         "PM" /* Trailing string 12:00 to 23:59  */
  "PM_National"  "sCurrency"     "$"  /* Currency Symbol string          */
  "PM_National"  "sThousand"     ","  /* Thousands separator string      */
  "PM_National"  "sDecimal"      "."  /* Decimal separator string        */
  "PM_National"  "sDate"         "-"  /* Date separator string           */
  "PM_National"  "sTime"         ":"  /* time separator string           */
  "PM_National"  "sList"         ","  /* List separator string.          */
  "PM_National"  "iMeasurement"  "2"  /* 1=English, 2=Metric,            */
                                      /* 3=Points, 4=Pica                */

  "PM_IBMBGA"    "ALTSYSFONT" "0"     /* For 8514 adapter only           */
  "PM_IBMBGA"    "FASTSS"     "0"     /* For 8514 adapter only           */

  "EPM"          "EPMIniPath" "\OS2\EPM.INI"
  "PMDiary"      "IniPath"    "\OS2\PMDIARY.INI"
/* Change default for Win-OS/2 Setup Object - R206 60555 */
  "WINOS2" "PM_GlobalWindows31Settings"
           "DPMI_MEMORY_LIMIT=64;PROGTYPE=PROG_31_STD;KBD_ALTHOME_BYPASS=1;
            VIDEO_SWITCH_NOTIFICATION=1;VIDEO_8514A_XGA_IOTRAP=0"

  "PM_Workplace:InstallGroups" "1" "1"
  "PM_Workplace:InstallDiskOnDesktop" "C" "ICONPOS=87 4"

  "PM_InstallObject" "Desktop;WPDesktop;?:\" "OBJECTID=<WP_DESKTOP>"

  /* remove PMDDE object from workplace */
  "PM_InstallObject" "Deleted;WPProgram;<WP_TOOLS>;DELETE"
                     "OBJECTID=<WP_PMDDE>"
END
```

Figure 80. CHNGDESK.RC National and Desktop Settings (Part 3 of 12)

```
STRINGTABLE REPLACEMODE
BEGIN
  "PM_InstallObject" "Nowhere;WPFolder;?:\"  "OBJECTID=<WP_NOWHERE>"
  "PM_InstallObject" "View;WPProgram;<WP_NOWHERE>;UPDATE"
                     "EXENAME=?:\OS2\VIEW.EXE;ASSOCFILTER=*.INF;
                     NOTVISIBLE=YES;OBJECTID=<WP_VIEWINF>"
  "PM_InstallObject" "Templates;WPTemplates;<WP_DESKTOP>"
                     "HELPPANEL=1568O;NODELETE=YES;ICONPOS=3 4;
                     OBJECTID=<WP_TEMPS>"
  "PM_InstallObject" "Minimized–Window Viewer;WPMinWinViewer;<WP_DESKTOP>"
                     "ICONPOS=4 2O;OBJECTID=<WP_VIEWER>"
  "PM_InstallObject" "Shredder;WPShredder;<WP_DESKTOP>"
                     "ICONPOS=93 4; OBJECTID=<WP_SHRED>"

  "PM_InstallObject" "OS/2 Window:1;WPProgram;<WP_DESKTOP>"
                     "EXENAME=*;PROGTYPE=WINDOWABLEVIO;NODELETE=YES;
                     ICONPOS=61 4;HELPPANEL=8O1O;OBJECTID=<WP_OS2WIN1>"
  "PM_InstallObject" "OS/2 Window:2;WPProgram;<WP_DESKTOP>"
                     "EXENAME=*;PROGTYPE=WINDOWABLEVIO;NODELETE=YES;
                     ICONPOS=71 4;HELPPANEL=8O1O;OBJECTID=<WP_OS2WIN2>"
  "PM_InstallObject" "DOS Window;WPProgram;<WP_DESKTOP>"
                     "EXENAME=*;PROGTYPE=WINDOWEDVDM;NODELETE=YES;
                     ICONPOS=8O 4;HELPPANEL=8O12;OBJECTID=<WP_DOSWIN1>"
  "PM_InstallObject" "Printer;PDView;<WP_DESKTOP>"
                     "Printer;OBJECTID=<WP_PDVIEW>"
  "PM_InstallObject" "Network;WPNetwork;<WP_DESKTOP>"
                     "HELPPANEL=3OOOO;NODELETE=YES;OBJECTID=<WP_NETWORK>"
```

Figure 81. CHNGDESK.RC Desktop Objects (Part 4 of 12)

```
"PM_InstallObject" "Information;WPFolder;<WP_DESKTOP>"
                   "HELPPANEL=13092;ICONRESOURCE=60 PMWP;ICONPOS=3 60;
                   NODELETE=YES;OBJECTID=<WP_INFO>"
  "PM_InstallObject" "Start Here;WPProgram;<WP_INFO>"
                     "EXENAME=STHR.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\HELP;
                     HELPPANEL=9278;OBJECTID=<WP_STHR>"
  "PM_InstallObject" "Master Help Index;Mindex;<WP_INFO>"
                     "INDEX=HELP;NODELETE=YES;OBJECTID=<WP_MINDEX>"
  "PM_InstallObject" "Glossary;Mindex;<WP_INFO>"
                     "INDEX=GLOSSARY;NODELETE=YES;OBJECTID=<WP_GLOSS>"
  "PM_InstallObject" "Command Reference;WPProgram;<WP_INFO>;UPDATE"
                     "EXENAME=?:\OS2\VIEW.EXE;PARAMETERS=CMDREF.INF;
                     HELPPANEL=9301;OBJECTID=<WP_CMDREF>"
  "PM_InstallObject" "REXX Information;WPProgram;<WP_INFO>;UPDATE"
                     "EXENAME=?:\OS2\VIEW.EXE;PARAMETERS=REXX.INF;
                     HELPPANEL=9300;OBJECTID=<WP_REXREF>"
  "PM_InstallObject" "ReadMe;WPShadow;<WP_INFO>;REPLACE"
                     "SHADOWID=?:\readme;OBJECTID=<WP_RDME>"
"PM_InstallObject" "OS/2 System;WPFolder;<WP_DESKTOP>"
                   "HELPPANEL=4002;NODELETE=YES;ICONRESOURCE=61 PMWP;
                   ICONPOS=3 70;ICONVIEWPOS=18 60 75 22;OBJECTID=<WP_OS2SYS>"
  "PM_InstallObject" "Drives;WPDrives;<WP_OS2SYS>"
                     "NODELETE=YES;OBJECTID=<WP_DRIVES>"
  "PM_InstallObject" "Startup;WPStartup;<WP_OS2SYS>"
                     "HELPPANEL=8002;NODELETE=YES;OBJECTID=<WP_START>"
```

Figure 82. CHNGDESK.RC Information and OS/2 System Objects (Part 5 of 12)

```
"PM_InstallObject" "System Setup;WPFolder;<WP_OS2SYS>" "HELPPANEL=1220;
                   ICONRESOURCE=59 PMWP;OBJECTID=<WP_CONFIG>"
  "PM_InstallObject" "System Clock;WPClock;<WP_CONFIG>"
                     "OBJECTID=<WP_CLOCK>"
  "PM_InstallObject" "Keyboard;WPKeyboard;<WP_CONFIG>" "OBJECTID=<WP_KEYB>"
  "PM_InstallObject" "Selective Install;WPProgram;<WP_CONFIG>;UPDATE"
                     "EXENAME=?:\OS2\INSTALL\INSTALL.EXE;
                     HELPPANEL=12220;OBJECTID=<WP_INST>"
  "PM_InstallObject" "Mouse;WPMouse;<WP_CONFIG>" "OBJECTID=<WP_MOUSE>"
  "PM_InstallObject" "Device Driver Install;WPProgram;<WP_CONFIG>;UPDATE"
                     "EXENAME=?:\OS2\INSTALL\DDINSTAL.EXE;
                     HELPPANEL=12221;OBJECTID=<WP_DDINST>"
  "PM_InstallObject" "Display Driver Install;WPProgram;<WP_CONFIG>;UPDATE"
                     "EXENAME=?:\OS2\INSTALL\DSPINSTL.EXE;
                     HELPPANEL=12221;OBJECTID=<WP_DSPINST>"
  "PM_InstallObject" ";WPWinConfig;<WP_CONFIG>" "OBJECTID=<WP_WINCFG>"
  "PM_InstallObject" "Migrate Applications;WPProgram;<WP_CONFIG>;UPDATE"
                     "EXENAME=?:\OS2\INSTALL\MIGRATE.EXE;
                     HELPPANEL=4508;OBJECTID=<WP_MIGAPP>"
  "PM_InstallObject" "Sound;WPSound;<WP_CONFIG>" "OBJECTID=<WP_SOUND>"
  "PM_InstallObject" "System;WPSystem;<WP_CONFIG>"
                     "HELPPANEL=9259;OBJECTID=<WP_SYSTEM>"
  "PM_InstallObject" "Country;WPCountry;<WP_CONFIG>" "OBJECTID=<WP_CNTRY>"
  "PM_InstallObject" "Font Palette;WPFontPalette;<WP_CONFIG>"
                     "NODELETE=YES;OBJECTID=<WP_FNTPAL>"
  "PM_InstallObject" "Color Palette;WPColorPalette;<WP_CONFIG>"
                     "NODELETE=YES;OBJECTID=<WP_CLRPAL>"
  "PM_InstallObject" "Scheme Palette;WPSchemePalette;<WP_CONFIG>"
                     "NODELETE=YES;AUTOSETUP=YES;OBJECTID=<WP_SCHPAL>"
  "PM_InstallObject" "Spooler;WPSpool;<WP_CONFIG>" "OBJECTID=<WP_SPOOL>"
  "PM_InstallObject" "Adobe Type Manager;WPProgram;<WP_CONFIG>;UPDATE"
                     "EXENAME=?:\OS2\MDOS\WINOS2\ATMCNTRL.EXE;
                     PROGTYPE=WINDOWEDWIN;OBJECTID=<WP_WINATM>"
  "PM_InstallObject" "Windows Control Panel;WPProgram;<WP_CONFIG>;UPDATE"
                     "EXENAME=?:\OS2\MDOS\WINOS2\CONTROL.EXE;
                     PROGTYPE=WINDOWEDWIN;OBJECTID=<WP_WINCONTROL>"
```

Figure 83. CHNGDESK.RC System Setup Objects (Part 6 of 12)

```
"PM_InstallObject" "Command Prompts;WPFolder;<WP_OS2SYS>"
                   "HELPPANEL=8008;OBJECTID=<WP_PROMPTS>"
  "PM_InstallObject" "OS/2 Full Screen;WPProgram;<WP_PROMPTS>"
                     "EXENAME=*;PROGTYPE=FULLSCREEN;HELPPANEL=8009;
                     OBJECTID=<WP_OS2FS>"
  "PM_InstallObject" "OS/2 Window;WPProgram;<WP_PROMPTS>"
                     "EXENAME=*;PROGTYPE=WINDOWABLEVIO;HELPPANEL=8010;
                     OBJECTID=<WP_OS2WIN>"
  "PM_InstallObject" "DOS Full Screen;WPProgram;<WP_PROMPTS>"
                     "EXENAME=*;PROGTYPE=VDM;HELPPANEL=8011;
                     OBJECTID=<WP_DOSFS>"
  "PM_InstallObject" "DOS Window;WPProgram;<WP_PROMPTS>"
                     "EXENAME=*;PROGTYPE=WINDOWEDVDM;HELPPANEL=8012;
                     OBJECTID=<WP_DOSWIN>"
 "PM_InstallObject" "WIN-OS/2 Full Screen;WPProgram;<WP_PROMPTS>"
                    "EXENAME=*;PROGTYPE=PROG_31_STD;
                    SET KBD_ALTHOME_BYPASS=1;
                    SET KBD_CTRL_BYPASS=CTRL_ESC;
                    SET VIDEO_SWITCH_NOTIFICATION=1;
                    SET VIDEO_8514A_XGA_IOTRAP=0;
                    SET DPMI_MEMORY_LIMIT=64;
                    HELPPANEL=8 022;OBJECTID=<WP_WINFS>"
  "PM_InstallObject" "DOS from Drive A:;WPProgram;<WP_PROMPTS>"
                     "EXENAME=*;PROGTYPE=VDM;SET DOS_STARTUP_DRIVE=A:;
                     HELPPANEL=8529;OBJECTID=<WP_DOS_DRV_A>"
  "PM_InstallObject" "Dual Boot;WPProgram;<WP_PROMPTS>"
                     "EXENAME=BOOT.COM;PROGTYPE=WINDOWABLEVIO;
                     PARAMETERS=/DOS;HELPPANEL=8029;OBJECTID=<WP_DBOOT>"
"PM_InstallObject" "Games;WPFolder;<WP_OS2SYS>"
                   "HELPPANEL=13091;OBJECTID=<WP_GAMES>"
  "PM_InstallObject" "Solitaire - Klondike;WPProgram;<WP_GAMES>;UPDATE"
                     "EXENAME=?:\OS2\APPS\KLONDIKE.EXE;PROGTYPE=PM;
                     HELPPANEL=20295;OBJECTID=<WP_KLDK>"
  "PM_InstallObject" "Cat and Mouse;WPProgram;<WP_GAMES>;UPDATE"
                     "EXENAME=?:\OS2\APPS\NEKO.EXE;PROGTYPE=PM;
                     HELPPANEL=20293;OBJECTID=<WP_NEKO>"
```

Figure 84. CHNGDESK.RC Command Prompts and Games Objects (Part 7 of 12)

```
"PM_InstallObject" "Productivity;WPFolder;<WP_OS2SYS>"
                   "HELPPANEL=13090;OBJECTID=<WP_TOOLS>"
  "PM_InstallObject" "Enhanced Editor;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=EPM.EXE;PROGTYPE=PM;HELPPANEL=20278;
                     OBJECTID=<WP_EPM>"
  "PM_InstallObject" "Seek and Scan Files;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMSEEK.EXE;PROGTYPE=PM;
                     HELPPANEL=20285;OBJECTID=<WP_SEEK>"
  "PM_InstallObject" "Icon Editor;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\ICONEDIT.EXE;PROGTYPE=PM;
                     ASSOCFILTER=*.BMP,*.PTR,*.ICO;HELPPANEL=20279;
                     OBJECTID=<WP_ICON>"
  "PM_InstallObject" "Pulse;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PULSE.EXE;PROGTYPE=PM;
                     HELPPANEL=20284;OBJECTID=<WP_PULSE>"
  "PM_InstallObject" "Calculator;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDCALC.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20272;
                     OBJECTID=<WP_DCALC>"
  "PM_InstallObject" "Notepad;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDNOTE.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20281;
                     OBJECTID=<WP_DNOTE>"
  "PM_InstallObject" "Alarms;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDALARM.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20270;
                     OBJECTID=<WP_DALARM>"
  "PM_InstallObject" "Calendar;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDCALEN.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20273;
                     OBJECTID=<WP_DCALEM>"
  "PM_InstallObject" "Planner Archive;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDDARC.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20282;
                     OBJECTID=<WP_DDARC>"
  "PM_InstallObject" "Daily Planner;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDDIARY.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20275;
                     OBJECTID=<WP_DDIARY>"
  "PM_InstallObject" "Activities List;WPProgram;<WP_TOOLS>;UPDATE"
                     "EXENAME=?:\OS2\APPS\PMDLIST.EXE;PROGTYPE=PM;
                     STARTUPDIR=\OS2\APPS;HELPPANEL=20271;
                     OBJECTID=<WP_DLIST>"
```

Figure 85. CHNGDESK.RC Productivity Objects 1 (Part 8 of 12)

```
      "PM_InstallObject" "Monthly Planner;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMDMONTH.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20280;
                         OBJECTID=<WP_DMNTH>"
      "PM_InstallObject" "To-Do List Archive;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMDTARC.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20290;
                         OBJECTID=<WP_DTARC>"
      "PM_InstallObject" "To-Do List;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMDTODO.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20289;
                         OBJECTID=<WP_TODO>"
      "PM_InstallObject" "Tune Editor;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMDTUNE.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20291;
                         OBJECTID=<WP_TUNE>"
      "PM_InstallObject" "Spreadsheet;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMSPREAD.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20287;
                         OBJECTID=<WP_SPREAD>"
      "PM_InstallObject" "Sticky Pad;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMSTICKY.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20288;
                         OBJECTID=<WP_STICKY>"
      "PM_InstallObject" "Database;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\APPS\PMMBASE.EXE;PROGTYPE=PM;
                         STARTUPDIR=\OS2\APPS;HELPPANEL=20276;
                         OBJECTID=<WP_DBASE>"
      "PM_InstallObject" "Clipboard Viewer;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\CLIPOS2.EXE;PROGTYPE=PM;
                         HELPPANEL=20274;OBJECTID=<WP_CLIPV>"
      "PM_InstallObject" "OS/2 System Editor;WPProgram;<WP_TOOLS>;UPDATE"
                         "EXENAME=?:\OS2\E.EXE;PROGTYPE=PM;HELPPANEL=9289;
                         ASSOCTYPE=Plain Text,OS/2 Command File,
                         DOS Command File,,;ASSOCFILTER=*.DOC,*.TXT,,;
                         OBJECTID=<WP_SYSED>"
  "PM_InstallObject" "Enhanced Editor;WPShadow;<WP_DESKTOP>;UPDATE"
                     "ICONPOS=12 4;SHADOWID=<WP_EPM>"
  "PM_InstallObject" "Drives;WPShadow;<WP_DESKTOP>"
                     "NODELETE=YES;ICONPOS=20 4;SHADOWID=<WP_DRIVES>"
END
```

Figure 86. CHNGDESK.RC Productivity Objects 2 and Shadows (Part 9 of 12)

```
CODEPAGE 850

STRINGTABLE
BEGIN

  "PM_Colors"          "Display"               "XGA"
  "PM_Colors"          "ActiveBorder"          "255 255 128"
  "PM_Default_Colors"  "ActiveBorder"          "255 255 128"
  "PM_Colors"          "ActiveTitle"           " 64 128 128"
  "PM_Default_Colors"  "ActiveTitle"           " 64 128 128"
  "PM_Colors"          "ActiveTitleText"       "255 255 255"
  "PM_Default_Colors"  "ActiveTitleText"       "255 255 255"
  "PM_Colors"          "ActiveTitleTextBgnd"   " 64 128 128"
  "PM_Default_Colors"  "ActiveTitleTextBgnd"   " 64 128 128"
  "PM_Colors"          "AppWorkspace"          "255 255 224"
  "PM_Default_Colors"  "AppWorkspace"          "255 255 224"
  "PM_Colors"          "Background"            "204 204 204"
  "PM_Default_Colors"  "Background"            "204 204 204"
  "PM_Colors"          "ButtonDark"            "128 128 128"
  "PM_Default_Colors"  "ButtonDark"            "128 128 128"
  "PM_Colors"          "ButtonDefault"         "  0   0   0"
  "PM_Default_Colors"  "ButtonDefault"         "  0   0   0"
  "PM_Colors"          "ButtonLight"           "255 255 255"
  "PM_Default_Colors"  "ButtonLight"           "255 255 255"
  "PM_Colors"          "ButtonMiddle"          "204 204 204"
  "PM_Default_Colors"  "ButtonMiddle"          "204 204 204"
  "PM_Colors"          "DialogBackground"      "204 204 204"
  "PM_Default_Colors"  "DialogBackground"      "204 204 204"
  "PM_Colors"          "EntryField"            "255 255 204"
  "PM_Default_Colors"  "EntryField"            "255 255 204"
  "PM_Colors"          "FieldBackground"       "204 204 204"
  "PM_Default_Colors"  "FieldBackground"       "204 204 204"
```

Figure 87. CHNGDESK.RC Color Settings 1 (Part 10 of 12)

```
"PM_Colors"          "HelpBackground"         "255 255 255"
"PM_Default_Colors"  "HelpBackground"         "255 255 255"
"PM_Colors"          "HelpHilite"             "  0 128 128"
"PM_Default_Colors"  "HelpHilite"             "  0 128 128"
"PM_Colors"          "HelpText"               "  0   0 128"
"PM_Default_Colors"  "HelpText"               "  0   0 128"
"PM_Colors"          "HiliteBackground"       " 96  96  96"
"PM_Default_Colors"  "HiliteBackground"       " 96  96  96"
"PM_Colors"          "HiliteForeground"       "255 255 255"
"PM_Default_Colors"  "HiliteForeground"       "255 255 255"
"PM_Colors"          "IconText"               "  0   0   0"
"PM_Default_Colors"  "IconText"               "  0   0   0"
"PM_Colors"          "InactiveBorder"         "204 204 204"
"PM_Default_Colors"  "InactiveBorder"         "204 204 204"
"PM_Colors"          "InactiveTitle"          "204 204 204"
"PM_Default_Colors"  "InactiveTitle"          "204 204 204"
"PM_Colors"          "InactiveTitleText"      "128 128 128"
"PM_Default_Colors"  "InactiveTitleText"      "128 128 128"
"PM_Colors"          "InactiveTitleTextBgnd"  "204 204 204"
"PM_Default_Colors"  "InactiveTitleTextBgnd"  "204 204 204"
"PM_Colors"          "Menu"                   "204 204 204"
"PM_Default_Colors"  "Menu"                   "204 204 204"
"PM_Colors"          "MenuText"               "  0   0   0"
"PM_Default_Colors"  "MenuText"               "  0   0   0"
"PM_Colors"          "MenuHilite"             "204 204 204"
"PM_Default_Colors"  "MenuHilite"             "204 204 204"
"PM_Colors"          "MenuHiliteText"         "  0   0   0"
"PM_Default_Colors"  "MenuHiliteText"         "  0   0   0"
"PM_Colors"          "MenuDisabledText"       "128 128 128"
"PM_Default_Colors"  "MenuDisabledText"       "128 128 128"
"PM_Colors"          "OutputText"             "  0   0   0"
"PM_Default_Colors"  "OutputText"             "  0   0   0"
"PM_Colors"          "PageBackground"         "255 255 255"
"PM_Default_Colors"  "PageBackground"         "255 255 255"
"PM_Colors"          "Scrollbar"              "192 192 192"
"PM_Default_Colors"  "Scrollbar"              "192 192 192"
```

Figure 88. CHNGDESK.RC Color Settings 2 (Part 11 of 12)

```
  "PM_Colors"          "Shadow"               "128 128 128"
  "PM_Default_Colors"  "Shadow"               "128 128 128"
  "PM_Colors"          "ShadowHiliteBgnd"     "128 128 128"
  "PM_Default_Colors"  "ShadowHiliteBgnd"     "128 128 128"
  "PM_Colors"          "ShadowHiliteFgnd"     "255 255 255"
  "PM_Default_Colors"  "ShadowHiliteFgnd"     "255 255 255"
  "PM_Colors"          "ShadowText"           "128 128 128"
  "PM_Default_Colors"  "ShadowText"           "128 128 128"
  "PM_Colors"          "TitleBottom"          "128 128 128"
  "PM_Default_Colors"  "TitleBottom"          "128 128 128"
  "PM_Colors"          "TitleText"            "255 255 255"
  "PM_Default_Colors"  "TitleText"            "255 255 255"
  "PM_Colors"          "Window"               "255 255 255"
  "PM_Default_Colors"  "Window"               "255 255 255"
  "PM_Colors"          "WindowFrame"          "128 128 128"
  "PM_Default_Colors"  "WindowFrame"          "128 128 128"
  "PM_Colors"          "WindowStaticText"     "  0   0 128"
  "PM_Default_Colors"  "WindowStaticText"     "  0   0 128"
  "PM_Colors"          "WindowText"           "  0   0   0"
  "PM_Default_Colors"  "WindowText"           "  0   0   0"
  "PM_DISPLAYDRIVERS"  "IBMXGA32"             "IBMXGA32"
  "PM_DISPLAYDRIVERS"  "CURRENTDRIVER"        "IBMXGA32"
  "PM_DISPLAYDRIVERS"  "DEFAULTDRIVER"        "IBMXGA32"
  ""                   ""                     ""
  "PM_DISPLAYDRIVERS"      "RESOLUTION_CHANGED"      "1"
END
```

Figure 89. CHNGDESK.RC Color Settings 3 and Display Drivers (Part 12 of 12)

3.7.2 Using an Existing Desktop

In the next examples we will be using an existing desktop and make changes to it. This means that the user .INI file must exist, and the .RC file must be compiled over the existing user .INI file. See 3.3, “The MAKEINI Compiler” on page 21 for more information on compiling the .RC files. We used the standard OS/2.INI file that was created after installation.

There are some objects on the desktop that cannot be modified *after* installation by compiling a .RC file over them. One of these objects is the Templates folder. The problem is that we cannot change the templates with the .RC files as they are created by the system and have no object ID. This makes it very difficult to reference these objects. Another folder that has the same problem is the Drives folder. The Drives objects are created by the system and are set, by default, non-deletable. The other setting we found that you cannot change by updating an .INI file with an .RC file is the desktop

name. It will always have the name that it was created with unless it is changed by editing the desktop name in the Desktop settings notebook.

3.7.2.1 Creating a Simple Desktop From an Existing Desktop

With SMALLINI.RC we wanted to create the same desktop we created with the SMALDESK.RC file in 3.7.1.1, "Creating a Simple Desktop" on page 88 which created a new desktop. Instead of not installing all the objects, normally installed with OS/2, as we have done with the SMALDESK.RC we are now going to remove all the objects from the desktop and then install the folders and objects that we need.

We start the SMALLINI.RC file with the .RC file header explained in 3.5.1, "RC File Header" on page 29. We then add the new associations for our data files. We change the default minimize button to minimize to the desktop instead of the minimized viewer as we are going to delete the Minimized Window Viewer. We then begin a second string table but with replace mode so that we can delete and change objects. Most of the objects in the System Setup folder are not deletable, and we have to update these objects to change them to be deletable. This is done by setting NODELETE=NO to each object. We then remove all the objects in the System Setup folder once by deleting the System folder. We do the same for all the other folders on the desktop.

The only desktop folders that we cannot delete are the Templates folder and the OS/2 System folder. The OS/2 Systems folder cannot be deleted because it contains the Drives folder, which contains the Drives objects which cannot be deleted. Another object we could not delete is the Drive A object as it is not installed with a PM_InstallObject application value, but with a PM_Workplace:InstallDiskOnDesktop application value.

The folders with their objects are then installed on the desktop. These folders are described in 3.7.1.1, "Creating a Simple Desktop" on page 88. Here is a summary of the SMALLINI.RC file:

- Create the associations for the data files.
- Set the default minimize behavior to minimize to desktop.
- Delete all the unwanted folders and objects.
- Create the four folders containing our data files and creating the hidden program objects.

The changed items described in the text are all in bold letters in the listings of SMALLINI.RC.

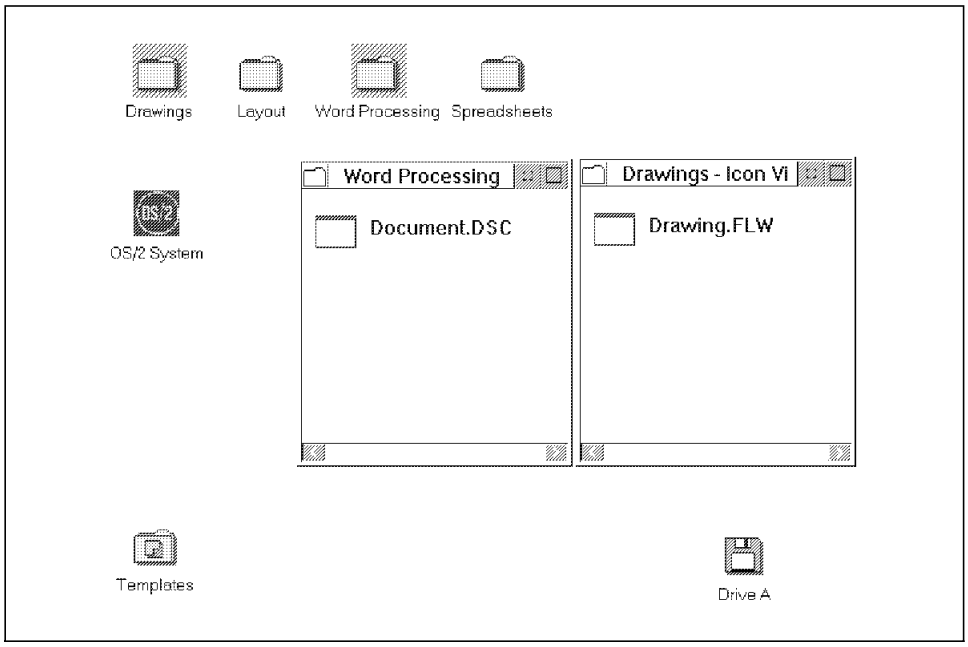


Figure 90. A Desktop Changed with SMALLINI.RC

```
/* File used by MAKEINI.EXE to update a user.INI                    */

CODEPAGE 850

STRINGTABLE
BEGIN

  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""

  "PM_ASSOC_TYPE"      "Spreadsheet"          ""
  "PM_ASSOC_TYPE"      "Wordprocessor"        ""
  "PM_ASSOC_TYPE"      "Drawings"             ""

  "PM_DefaultSetup" "MINWIN" "DESKTOP"


END
```

Figure 91. SMALLINI.RC File Header Associations (Part 1 of 4)

```
STRINGTABLE REPLACEMODE
BEGIN
  "PM_InstallObject" "Minimized-Window Viewer;WPMinWinViewer;
                      <WP_DESKTOP>;DELETE"
                     "OBJECTID=<WP_VIEWER>"
  "PM_InstallObject" "Shredder;WPShredder;<WP_DESKTOP>;DELETE"
                     "OBJECTID=<WP_SHRED>"
    "PM_InstallObject" "Glossary;Mindex;<WP_INFO>;UPDATE"
                       "INDEX=GLOSSARY;NODELETE=NO;OBJECTID=<WP_GLOSS>"
  "PM_InstallObject" "Information;WPFolder;<WP_DESKTOP>;DELETE"
                     "OBJECTID=<WP_INFO>"
    "PM_InstallObject" "Start Here;WPProgram;<WP_DESKTOP>;DELETE"
                       "EXENAME=STHR.EXE;PROGTYPE=PM;
                       OBJECTID=<WP_STHR>"
    "PM_InstallObject" "Master Help Index;Mindex;<WP_DESKTOP>;DELETE"
                       "NODELETE=NO;OBJECTID=<WP_MINDEX>"
    "PM_InstallObject" "Startup;WPStartup;<WP_OS2SYS>;DELETE"
                       "HELPPANEL=8002;NODELETE=NO;OBJECTID=<WP_START>"
      "PM_InstallObject" "System Clock;WPClock;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_CLOCK>"
      "PM_InstallObject" "Keyboard;WPKeyboard;<WP_CONFIG>;DELETE"
                         "NODELETE=NO;OBJECTID=<WP_KEYB>"
      "PM_InstallObject" "Mouse;WPMouse;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_MOUSE>"
      "PM_InstallObject" ";WPWinConfig;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_WINCFG>"
      "PM_InstallObject" "Sound;WPSound;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_SOUND>"
      "PM_InstallObject" "System;WPSystem;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_SYSTEM>"
      "PM_InstallObject" "Country;WPCountry;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_CNTRY>"
      "PM_InstallObject" "Font Palette;WPFontPalette;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_FNTPAL>"
      "PM_InstallObject" "Color Palette;WPColorPalette;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_CLRPAL>"
      "PM_InstallObject" "Scheme Palette;WPSchemePalette;<WP_CONFIG>;UPDATE"
                         "NODELETE=NO;OBJECTID=<WP_SCHPAL>"
      "PM_InstallObject" "Spooler;WPSpool;<WP_CONFIG>;UPDATE"
                          "NODELETE=NO;OBJECTID=<WP_SPOOL>"
      "PM_InstallObject" ";WPPower;<WP_CONFIG>;UPDATE"
                          "NODELETE=NO;OBJECTID=<WP_POWER>"
    "PM_InstallObject" "Command Prompts;WPFolder;<WP_OS2SYS>;DELETE"
                       "OBJECTID=<WP_PROMPTS>"
    "PM_InstallObject" "Games;WPFolder;<WP_OS2SYS>;DELETE"
                       "OBJECTID=<WP_GAMES>"
    "PM_InstallObject" "Productivity;WPFolder;<WP_OS2SYS>;DELETE"
                       "OBJECTID=<WP_TOOLS>"
```

Figure 92. SMALLINI.RC Delete Objects (Part 2 of 4)

```
"PM_InstallObject" "DeScribe;WPProgram;<WP_DESKTOP>"
                   "EXENAME=?:\DESCRIBE\DESCRIBE.EXE;PROGTYPE=PM;
                   NOTVISIBLE=YES;ASSOCTYPE=Wordprocessor,,;
                   ASSOCFILTER=*.DSC,,;OBJECTID=<WP_DESCRIBE>"
  "PM_InstallObject" "Lotus 1-2-3;WPProgram;<WP_DESKTOP>"
                   "EXENAME=?:\123W\123W.EXE;PROGTYPE=WINDOWEDWIN;
                   NOTVISIBLE=YES;ASSOCTYPE=Spreadsheet,,;
                   ASSOCFILTER=*.WK?,,;OBJECTID=<WP_LOTUS>"
  "PM_InstallObject" "FreeLance;WPProgram;<WP_DESKTOP>"
                   "EXENAME=?:\FL\FL.EXE;PROGTYPE=WINDOWEDWIN;
                   NOTVISIBLE=YES;ASSOCTYPE=Drawing,,;
                   ASSOCFILTER=*.FLW,,;OBJECTID=<WP_FL>"
  "PM_InstallObject" "Layout;WPFolder;<WP_DESKTOP>"
                   "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                   NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                   ICONVIEWPOS=20 50 60 20;OBJECTID=<WP_LAYOUT>"
    "PM_InstallObject" "Document.DSC;WPDataFile;<WP_LAYOUT>"
                     "TEMPLATE=YES;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                     NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Wordprocessor,,;
                     OBJECTID=<WP_TEMPWORD>"
    "PM_InstallObject" "Sheet.WK3;WPDataFile;<WP_LAYOUT>"
                     "TEMPLATE=YES;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                     NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Spreadsheet,,;
                     OBJECTID=<WP_TEMPSHEET>"
    "PM_InstallObject" "Drawing.FLW;WPDataFile;<WP_LAYOUT>"
                     "TEMPLATE=YES;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                     NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Drawing,,;
                     OBJECTID=<WP_TEMPDRAW>"
```

Figure 93. SMALLINI.RC Install Desktop and Template Objects (Part 3 of 4)

```
"PM_InstallObject" "Word Processing;WPFolder;<WP_DESKTOP>"
                   "ICONVIEW=FLOWED;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                   NODRAG=YES;NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                   ICONVIEWPOS=33 30 30 50;OBJECTID=<WP_WORDFOLDER>"
   "PM_InstallObject" "Document.DSC;WPDataFile;<WP_WORDFOLDER>"
                      "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                      NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Wordprocessor,,;
                      OBJECTID=<WP_WORDFILE>"
  "PM_InstallObject" "Spreadsheets;WPFolder;<WP_DESKTOP>"
                   "ICONVIEW=FLOWED;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                   NODRAG=YES;NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                   ICONVIEWPOS=1 30 30 50;OBJECTID=<WP_SHEETFOLDER>"
    "PM_InstallObject" "Sheet.WK3;WPDataFile;<WP_SHEETFOLDER>"
                      "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                      NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Spreadsheet,,;
                      OBJECTID=<WP_SHEETFILE>"
  "PM_InstallObject" "Drawings;WPFolder;<WP_DESKTOP>"
                   "ICONVIEW=FLOWED;NODELETE=YES;NOCOPY=YES;NOMOVE=YES;
                   NODRAG=YES;NOLINK=YES;NOSHADOW=YES;NORENAME=YES;
                   ICONVIEWPOS=66 30 30 50;OBJECTID=<WP_DRAWFOLDER>"
    "PM_InstallObject" "Drawing.FLW;WPDataFile;<WP_DRAWFOLDER>"
                      "NODELETE=YES;NOCOPY=YES;NOMOVE=YES;NODRAG=YES;
                      NOLINK=YES;NOSHADOW=YES;ASSOCTYPE=Drawing,,;
                      OBJECTID=<WP_DRAWFILE>"
END
```

Figure 94. SMALLINI.RC Install Folder Objects (Part 4 of 4)

3.7.2.2 Adding Objects to an Existing Desktop

In this example we are changing and adding objects to an existing desktop. We are also registering another font in the system. This font is called BlackChancery and consists of two files. The two files are BLACKCHA.OFM and BLACKCHA.PFB, and were copied from a previously installed desktop. Both these files were then copied to the PSFONTS directory on the boot drive of the system we are updating. If these files are not present in the /PSFONTS directory then the line in the .RC file will be ignored and the font will not be registered with the user. INI file.

The next string mode is in replace mode to change and add the objects. Most of the objects already exist, and we only changed their position. If an object's *location* is changed then it must be replaced with a REPLACE option; otherwise we can use the UPDATE option. Here is a list of what we are doing in the CHANGINI.RC file:

- We add the BlackChancery font
- We change the positions of the current folders and objects on the desktop
- We made some of the folders and objects non-deletable
- We add three command prompts windows:
  - OS/2 Window:1
  - OS/2 Window:2
  - DOS Window
- We moved the Start Here and Master Help Index object from the desktop to the Information folder
- We created additional setup icons in the System Setup folder for:
  - Display Drivers Install
  - Adobe Type Manager for Windows
  - Windows Control Panel
- We created the following two shadows on the desktop:
  - Enhanced Editor
  - Drives

The items described in the text are all in bold letters in the listings of CHANGINI.RC.

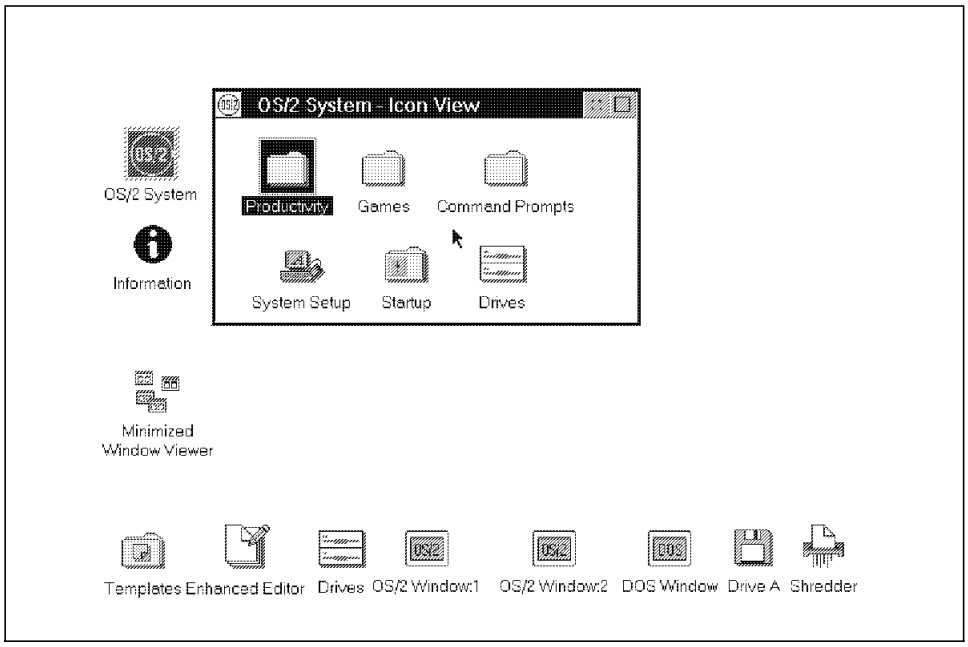


Figure 95. Objects Added to a Desktop With CHANGINI.RC

```
/* File used by MAKEINI.EXE to update a user.INI                        */

STRINGTABLE
BEGIN

  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""

  "PM_Fonts" "BLACKCHA.OFM" "\PSFONTS\BLACKCHA.OFM"
END
```

Figure 96. CHANGINI.RC File Header and Objects (Part 1 of 2)

```
STRINGTABLE REPLACEMODE
BEGIN
  "PM_InstallObject" "Templates;WPTemplates;<WP_DESKTOP>;UPDATE"
                     "NODELETE=YES;ICONPOS=4 5;OBJECTID=<WP_TEMPS>"
  "PM_InstallObject" "Minimized-Window Viewer;WPMinWinViewer;
                      <WP_DESKTOP>;UPDATE"
                     "ICONPOS=5 30;OBJECTID=<WP_VIEWER>"
  "PM_InstallObject" "Shredder;WPShredder;<WP_DESKTOP>;UPDATE"
                     "ICONPOS=90 5;OBJECTID=<WP_SHRED>"
  "PM_InstallObject" "OS/2 Window:1;WPProgram;<WP_DESKTOP>;REPLACE"
                     "EXENAME=*;PROGTYPE=WINDOWABLEVIO;NODELETE=YES;
                     ICONPOS=45 5;HELPPANEL=8010;OBJECTID=<WP_OS2WIN1>"
  "PM_InstallObject" "OS/2 Window:2;WPProgram;<WP_DESKTOP>;REPLACE"
                     "EXENAME=*;PROGTYPE=WINDOWABLEVIO;NODELETE=YES;
                     ICONPOS=60 5;HELPPANEL=8010;OBJECTID=<WP_OS2WIN2>"
  "PM_InstallObject" "DOS Window;WPProgram;<WP_DESKTOP>;REPLACE"
                     "EXENAME=*;PROGTYPE=WINDOWEDVDM;NODELETE=YES;
                     ICONPOS=70 5;HELPPANEL=8012;OBJECTID=<WP_DOSWIN1>"
  "PM_InstallObject" "Information;WPFolder;<WP_DESKTOP>;UPDATE"
                     "ICONPOS=4 55;NODELETE=YES;OBJECTID=<WP_INFO>"
    "PM_InstallObject" "Start Here;WPProgram;<WP_INFO>;REPLACE"
                       "EXENAME=STHR.EXE;PROGTYPE=PM;STARTUPDIR=\OS2\HELP;
                       HELPPANEL=9278;OBJECTID=<WP_STHR>"
    "PM_InstallObject" "Master Help Index;Mindex;<WP_INFO>;REPLACE"
                       "INDEX=HELP;NODELETE=YES;OBJECTID=<WP_MINDEX>"
  "PM_InstallObject" "OS/2 System;WPFolder;<WP_DESKTOP>;UPDATE"
                     "NODELETE=YES;ICONPOS=4 70;ICONVIEWPOS=9 59 35 23;
                     OBJECTID=<WP_OS2SYS>"
    "PM_InstallObject" "Display Driver Install;WPProgram;<WP_CONFIG>"
                       "EXENAME=?:\OS2\INSTALL\DSPINSTL.EXE;
                       OBJECTID=<WP_DSPINST>"
    "PM_InstallObject" "Adobe Type Manager;WPProgram;<WP_CONFIG>"
                       "EXENAME=?:\OS2\MDOS\WINOS2\ATMCNTRL.EXE;
                       PROGTYPE=WINDOWEDWIN;OBJECTID=<WP_WINATM>"
    "PM_InstallObject" "Windows Control Panel;WPProgram;<WP_CONFIG>"
                       "EXENAME=?:\OS2\MDOS\WINOS2\CONTROL.EXE;
                       PROGTYPE=WINDOWEDWIN;OBJECTID=<WP_WINCONTROL>"
    "PM_InstallObject" "Enhanced Editor;WPProgram;<WP_TOOLS>;UPDATE"
                       "EXENAME=EPM.EXE;PROGTYPE=PM;OBJECTID=<WP_EPM>"
  "PM_InstallObject" "Enhanced Editor;WPShadow;<WP_DESKTOP>;UPDATE"
                     "ICONPOS=15 5;SHADOWID=<WP_EPM>"
  "PM_InstallObject" "Drives;WPShadow;<WP_DESKTOP>"
                     "NODELETE=YES;ICONPOS=25 5;SHADOWID=<WP_DRIVES>"
END
STRINGTABLE
```

Figure 97. CHANGINI.RC Objects (Part 2 of 2)

Chapter 4. Using .INI files to Change the Desktop

If there is one thing that OS/2 users have been told to leave alone it is the OS/2 .INI files, and for good reason. If you have ever had a corrupt .INI file on your system you know what we mean. The aim of this chapter is to show you how you can safely configure the Workplace Shell using the .INI files.

In an operating system such as OS/2 there is a need to store certain information about the way the system is configured. This information could be such things as system colors and fonts or even which printers are installed and which printer drivers they use. In OS/2 2.1 a lot of this information is stored in two files called OS2.INI and OS2SYS.INI, which reside in the OS2 directory on the drive where OS/2 is installed.

If you were to look at an .INI file with a text editor you would not be able to make much sense of it. This is because it has been encoded. The reason for this is simple, speed. By encoding the .INI files the way they have the OS/2 developers have been able to give the Workplace Shell fast access to all the information it needs to run. This also means that in order for the rest of us to access the .INI files, we are going to have to use a method of encoding and decoding the data we wish to place there. Thankfully there are interfaces through MAKEINI, REXX, and C/C++ to allow us to do this.

The following information does not represent a complete reference for every piece of data that can be found in the .INI files. The intention has been to document only those settings that relate to the configuration of the Workplace Shell.

Warning

A word of warning - there is a lot of information in the .INI files that is critical to the way the Workplace Shell functions. This information was only meant to be used by the shell itself. Inserting information into the wrong places in the .INI files can have serious consequences for your system.

4.1 The Structure of a .INI File

Each .INI file is broken up into a number of sections called applications. Each application is then further broken up into a number of keys. Pieces of data, called key values, are then associated with each of the key names. For example, the system colors are stored in an application called PM_Colors. Within PM_Colors there are a number of keys relating to parts of the system we are able to change the color of. If we were to look at the key name IconText we might find the key value "0 0 0 " which tells OS/2 to make the color of icon text black.

4.1.1 The Difference Between OS2.INI and OS2SYS.INI

The question you are probably asking now is: "Why have two files?" The answer is that they perform slightly different functions. Each file has the same structure, that is it contains applications and keys, but there are different applications in each file.

The OS2.INI file is sometimes called the user .INI file. This is because it contains information that a user of OS/2 might want to change. Things like colors, country settings, mouse settings and some data about objects are stored here.

The OS2SYS.INI file, also called the system .INI file, tends to store information about the system you are running on. Things such as the color values for all of the default schemes and the configuration of printers attached to the system are stored here.

In this chapter we will concentrate on the user .INI file (that is, OS2.INI) as this is the file where the Workplace Shell stores most of its data.

4.2 Reading and Writing Data to the .INI Files

The first thing to remember about reading from and writing data to the .INI files is that it involves reading and writing whole key values. If you wish to change only a part of a key value you must typically read the whole value, change the parts that you want, and then write the whole value back to the .INI file you are working with. Similarly, if you want to look at only a part of a key value you must first read the whole value and then extract the part that you want.

The second thing to remember is that some of the data in the .INI files is formatted in such a way that it is easiest to access from C. This means that

REXX users might have to do a little manipulation of the data to get it in a form that the .INI files understand. In most cases all this means is placing or removing a special character (a null character) at the end of the data you wish to write.

The method used to access .INI files from REXX is to call the REXX utility function SysIni. This function is provided as part of the RexxUtil package of functions that comes with OS/2. By default the RexxUtil functions are not registered to REXX so the first thing that you must do in your program is to register SysIni (See Figure 98 and Figure 99). Alternatively if you are using other RexxUtil functions you may want to register all of them instead. We will use the second simpler form of registration shown in Figure 99 for the rest of the REXX examples in this chapter.

```
/*--------------------Checking and Registering SysIni-----------------*/
if RxFuncQuery('SysIni') = 1 then do
    rc = RxFuncAdd('SysIni', 'RexxUtil', 'SysIni')
    If rc = 0 then
      say 'SysIni registered OK'
    else
      say 'SysIni registration failed'
    end
else
  say 'SysIni already registered'
exit
```

Figure 98. Checking for and/or Registering SysIni

```
/*--------------------Simple Registration of SysIni--------------------*/
call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'
exit
```

Figure 99. Simply Registering SysIni

Notes

REXX must have been installed on the workstation(s) you are running these REXX programs on.

Not all applications and keys may be found in the .INI files after an initial system installation. Some are only registered if you change the defaults through the various settings notebooks. This will not cause a problem in writing data to the .INI files, but it can cause an error if you try to read an application or key that is not there. You should consider this when constructing your programs.

The SysIni function can be used in a number of ways to query and set data in the .INI files. SysIni can be called to:

- Retrieve the names of all applications
- Retrieve all keys for a single application
- Delete an application and all associated keys
- Delete a single key
- Retrieve a single key value
- Set a single key value

Of these we are most interested in retrieving and setting a single key value as well as retrieving all keys for a single application. For further information on SysIni, please consult the REXX Information which can be found inside the Information icon on the default Workplace Shell desktop.

4.2.1 Retrieving All Keys for a Single Application

To retrieve all keys for a single application it is necessary to pass SysIni the application name we are interested in as well as the name of a REXX stem to store the results. See Figure 100 on page 123 for an example of a REXX program that retrieves all of the key names for the application PM_ControlPanel which stores many of the system settings.

The application names we will be typically interested in doing this for are as follows:

- ClockProgram 2.2 in WP3
- Lock Up Workplace
- PM_AlternateInputMethods
- PM_Colors

- PM_ControlPanel
- PM_Default_Colors
- PM_Default_National
- PM_Fonts
- PM_National
- PM_SystemFonts
- PM_Workplace

```
/*--------------------------------------------------------------------
 * QUERYKEY.CMD List all key names for the application PM_ControlPanel
 * and writes them to the screen.
 *-------------------------------------------------------------------*/
/* initialize variables and register SysIni                         */
keynames. = ''                              /* stem to hold key names*/
appname   = 'PM_ControlPanel'               /* application name      */
inifile   = 'USER'                          /* .INI file to use      */
call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'     /* Register SysIni*/
/* query key names */
result = SysIni(inifile, appname, 'ALL:', 'keynames')
if result = 'ERROR:' then
  do
    say 'Error obtaining .INI file information'
    exit
  end
/* write out results */
say 'Key names for application ' || appname || ' are:'
do i = 1 to keynames.0
  say keynames.i
end
exit
```

Figure 100. Querying Key Names for an Application

The results of running this program on a typical OS/2 system are shown in Figure 101 on page 124. Please note that the results will vary depending on which system settings you have changed on your system. If you are getting error messages it may mean that your system is using the default values and does not have the information in the .INI file yet. If this is the case try opening the Keyboard object in the System Setup folder, changing something such as the cursor blink rate and then closing the Keyboard object to save the settings.

```
Key names for application PM_ControlPanel are:
ConfirmDelete
ConfirmSubDelete
ConfirmRenameFilesWithExt
ConfirmCopyMoveEtc
DisplayProgressInd
NameClash
MinButtonType
HiddenMinWindows
ContextMenuKB
TextEditKB
DoubleClickSpeed
MouseTrackingSpeed
SwapMouseButtons
Beep
ContextMenuMouse
TextEditMouse
BeginDragMouse
TaskListMouseAccess
EndDragMouse
CursorBlinkRate
KeyRepeatRate
KeyRepeatDelay
LogoDisplayTime
PrintScreen
Animation
BorderWidth
```

Figure 101. Results of Running QUERYKEY.CMD

4.2.2 Retrieving a Single Key Value

To retrieve a single key value SysIni requires only the application and key name as well as the .INI file to use. See Figure 102 for an example of a REXX program to retrieve the cursor blink rate setting. Note the code to remove the null character at the end of the key value.

```
/*--------------------------------------------------------------------
 * QUERYVAL.CMD query the key value for the application
 * PM_ControlPanel and key name CursorBlinkRate.
 *-------------------------------------------------------------------*/
/* initialize variables                                              */
keyname   = 'CursorBlinkRate'                    /* key name         */
appname   = 'PM_ControlPanel'                    /* application name */
inifile   = 'USER'                               /* INI file to use  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni  */
value = SysIni(inifile, appname, keyname)
if value = 'ERROR:' then
  do
    say 'Error obtaining .INI file information'
    exit
  end
/* remove terminating null character */
len = length(value)
value = left(value, len - 1)
/* write out results */
say 'Key value for application ',
     || appname || ' key name ' || keyname || ' is:'
say value
exit
```

Figure 102. Retrieving a Single Key Value

The values returned by this program are between 0 and 890. If no value is being returned try setting the cursor blink rate manually using the Keyboard settings in the system setup folder and closing the Keyboard settings before running the program again.

4.2.3 Setting a Single Key Value

The SysIni call to set a single key value is similar to the call to retrieve a value. To accomplish this we simply supply a value as an extra parameter. See Figure 103 for an example of how to do this. In this program we set the cursor blink rate to 0, its fastest setting. Note the addition of a trailing null character (equal to 00 in hexadecimal) to the end of the value to tell OS/2 that this value has ended.

```
/*--------------------------------------------------------------------
 * SETVAL.CMD Set the value of the CursorBlinkRate key in the
 * PM_ControlPanel application to 0.
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'CursorBlinkRate'                    /* key name         */
appname   = 'PM_ControlPanel'                    /* application name */
inifile   = 'USER'                               /* INI file to use  */
value     = '0'                                  /* value to insert  */
NULL      = '00'x                                /* null character   */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni  */
value = value || NULL                            /* add null         */
result = SysIni(inifile, appname, keyname, value)/* set key value    */
if result = 'ERROR:' then
  say 'Error setting .INI file information'
else
  say 'Key value successfully set to ' || value
exit
```

Figure 103. Setting a Single Key Value

Important

When you change *most* values in the .INI files you must shut down your system in order to see the changes take effect. The reason for this is that the Workplace Shell keeps its own copy of much of the information in the .INI files. By shutting down you force the Workplace Shell to read a new copy of what is in the .INI files. When you update settings through the settings notebooks the Workplace Shell updates its own internal copy of the data and then writes that data to the .INI file. That is how you see the changes immediately.

4.3 Changing the Desktop Settings

As was discussed in the introduction to this chapter there are many applications in the .INI files which are intended for use by the Workplace Shell alone. There are, however, a number of interesting and useful things you can do using the .INI files. Now that we have covered the basics of how to manipulate data in the INI files we will go on to some of the things you might want to do.

4.3.1 System Settings

Most of the settings for the Workplace Shell can be manipulated by using the key values under the application PM_ControlPanel. The formats of the key values vary and so do the steps required to write them to the .INI files. Values that are in Integer or Character format simply require the addition of a terminating null to the REXX variable value. Binary values have been supplied in tables where they occur to simplify their manipulation from REXX. For C programmers only, the types of binary values have been supplied to enable them to be written more easily from C.

NOTE

Unless otherwise specified, you must reboot the workstation to see the changes to the .INI file take effect.

The system confirmation key values control whether or not the Workplace Shell will ask you if you are sure before doing something. All of these values can be set manually from the Confirmations page of the System settings (see Figure 104 on page 128) located in the System Setup folder. In all there are five settings that you can change (see Table 26 on page 128).

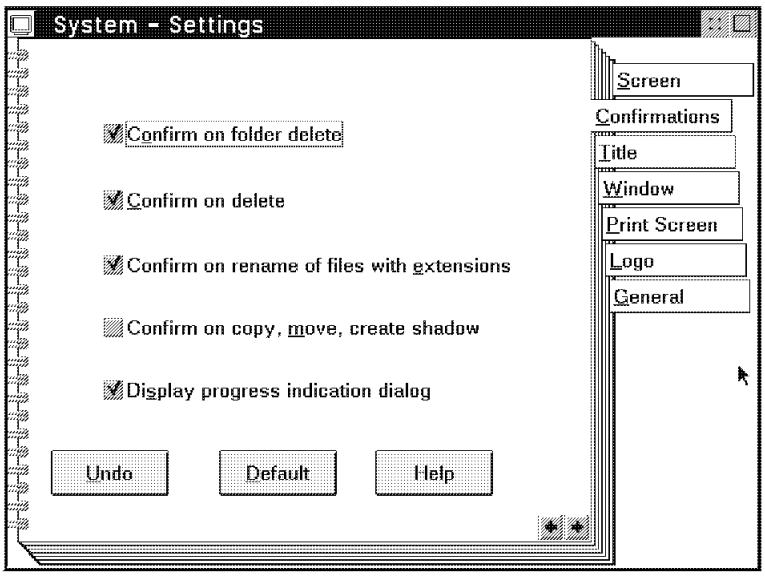


Figure 104. The Confirmations Page from the System Settings

Table 26. System Confirmation Key Names	
ConfirmCopyMoveEtc	Controls confirmation of a copy, move, or shadow operation on an object
ConfirmDelete	Controls confirmation of the deletion of an object
ConfirmRenameFilesWithExt	Controls confirmation of the rename of a file with an extension that changes the extension
ConfirmSubDelete	Controls confirmation of the deletion of a folder that contains other folders (subfolders)
DisplayProgressInd	Controls the display of the progress indication dialog that tells you when a copy, move, or delete operation is in progress.

The key values for each of these five keys can be either 1 to tell the Workplace Shell to display the confirmation dialog or 0 to disable the dialog (see Table 27 on page 129).

Table 27. Values for System Confirmations	
Value	**Meaning**
1	The Workplace Shell will display the confirmation dialog
0	The Workplace Shell will not display the confirmation dialog

For example, if you wanted to tell the Workplace Shell to ask you for confirmation when you move, copy, or shadow an object you could use the following REXX statements:

```
/*-------------------------------------------------------------------
 * Turn on confirmation of copy, move and shadow operations
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'ConfirmCopyMoveEtc'                /* key name        */
appname   = 'PM_ControlPanel'                   /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '1'                                 /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Key Name Animation
Format Binary
C type BOOL
Description This value controls the animation of windows in the Workplace Shell and corresponds to the "Animation" radio buttons on the Window page of the System settings (See Figure 114 on page 139). The possible values are as follows:

Table 28. Animation Setting Values

Value in Hex	Meaning
01000000	Animation is enabled
00000000	Animation is disabled

```
/*--------------------------------------------------------------------
 * Disable Animation
 *-------------------------------------------------------------------*/
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
result = SysIni('USER',,                         /* INI file        */
                'PM_ControlPanel',,           /* Application Name*/
                'Animation',,                    /* Key Name        */
                '00000000'x)                     /* Key Value       */
exit
```

Figure 105. Sample REXX to Disable Animation

Key Name Beep
Format Integer
Description This value corresponds to the "Warning beep" check box on the Warning Beep page of the Sound settings (see Figure 107 on page 132) and controls whether or not the system beeps when it brings up a warning. Possible values are as follows:

Table 29. Values for Beep

Value	Description
1	System beeps are enabled
0	System beeps are disabled

```
/* Disable Warning Beeps                                          */
/* initialize variables */
keyname   = 'Beep'                              /* key name        */
appname   = 'PM_ControlPanel'                   /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '0'                                 /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 106. Sample REXX to Disable Warning Beeps

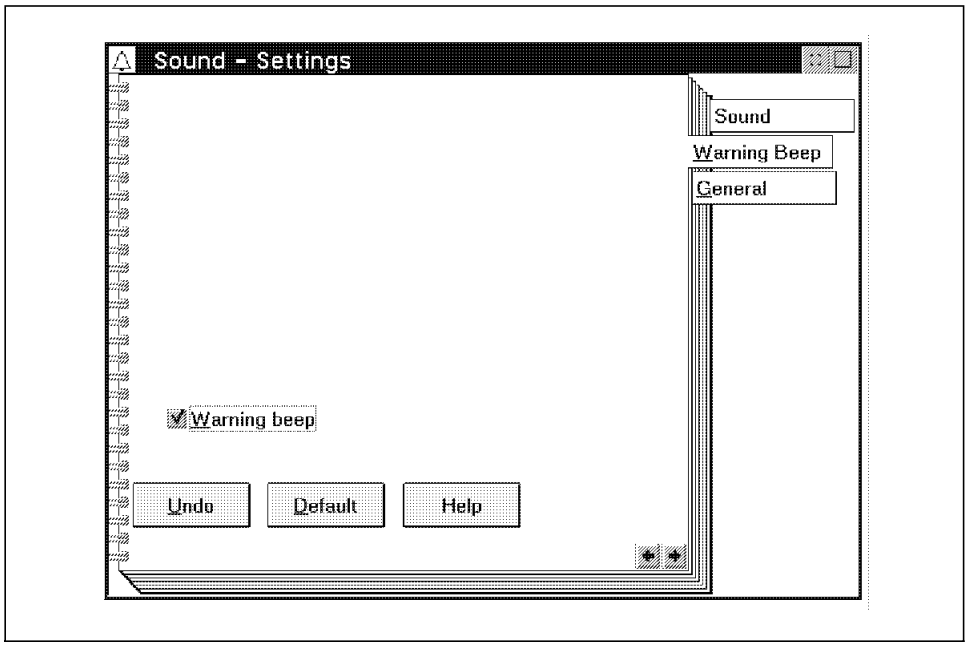


Figure 107. The Warning Beep Page from the Sound Settings

Key Name BorderWidth
Format Integer
Description This key value controls the default border width in the system. The way to set this is by editing the scheme you are using in the Scheme Palette object, which is in the System Setup folder. You can set the borderwidth to be what ever value you wish and then do an Alt drag of the scheme to your desktop and the system's default will be changed. The key value is the default border width in pixels.

```
/*--------------------------------------------------------------------
 * Set default border width to 10 pixels
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'BorderWidth'                        /* key name        */
appname   = 'PM_ControlPanel'                    /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '10'                                 /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 108. Sample REXX to Set the Default Border Width

Key Name CursorBlinkRate
Format Integer
Description This value corresponds to the ″Cursor blink rate″ setting on the Timing page of the Keyboard settings (see Figure 110 on page 135). This value relates to the amount of time between blinks of the cursor and can be between 890, for the slowest cursor blink rate, and 0, for the fastest cursor blink rate.

```
/*--------------------------------------------------------------------
 * Set cursor blink rate to 100
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'CursorBlinkRate'                    /* key name        */
appname   = 'PM_ControlPanel'                    /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '100'                                /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 109. Sample REXX to Set the Cursor Blink Rate

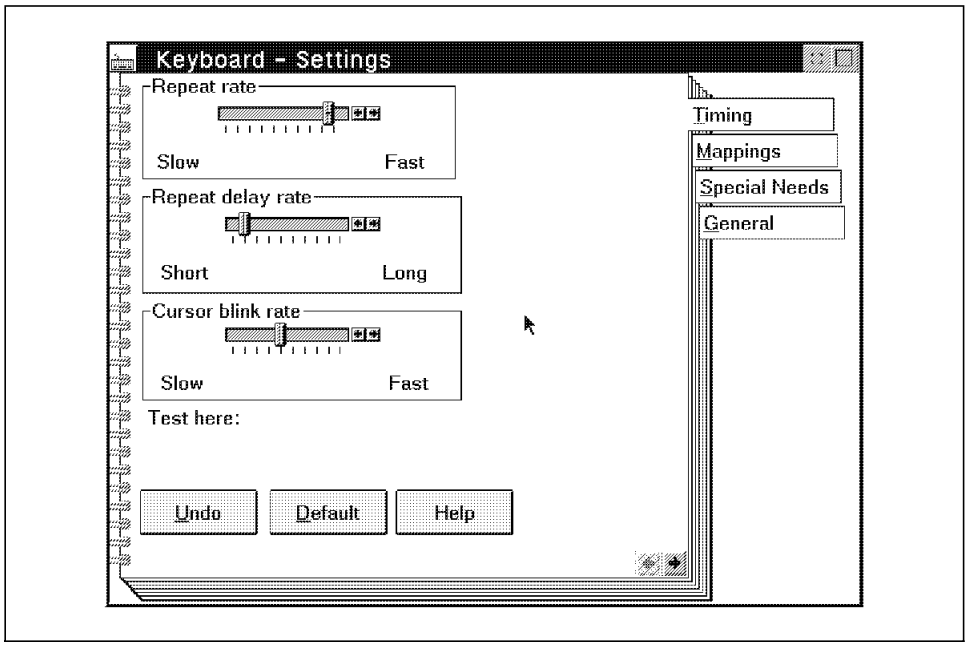


Figure 110. The Timing Page from the Keyboard Settings

Key Name KeyRepeatDelay
Format Integer
Description This value corresponds to the ″Repeat delay rate″ setting on the Timing page of the Keyboard settings (see Figure 110 on page 135) and controls how long a key must normally be held down in order for it to begin repeating (see also 4.3.8, “Workplace Shell Setup for Users with Special Needs” on page 189). The value can be between 0 for minimum repeat delay, and 890 for maximum repeat delay.

```
/*--------------------------------------------------------------------
 * Set keyboard repeat delay to 149
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'KeyRepeatDelay'                    /* key name        */
appname   = 'PM_ControlPanel'                   /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '149'                               /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 111. Sample REXX to Set the Keyboard Repeat Delay

Key Name KeyRepeatRate
Format Integer
Description This value corresponds to the ″Repeat rate″ setting on the Timing page of the Keyboard settings (see Figure 110 on page 135) and controls how fast characters appear when a key on the keyboard is repeating (See also 4.3.8, “Workplace Shell Setup for Users with Special Needs” on page 189). The value can be between 1 for minimum repeat rate and 20 for maximum repeat rate.

```
/*--------------------------------------------------------------------
 * Set keyboard repeat rate to 11
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'KeyRepeatRate'                     /* key name        */
appname   = 'PM_ControlPanel'                   /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '11'                                /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 112. Sample REXX to Set the Keyboard Repeat Rate

Key Name HiddenMinWindows
Format Integer
Description This value corresponds to the "Minimize button behavior" radio buttons on the Window page of the System settings (see Figure 114 on page 139) and controls what the Workplace Shell does when the minimize button is selected on an application or folder. Values are as follows:

Table 30. Minimize Behavior Values

Value	Corresponding radio button
1	Hide window
2	Minimize window to viewer
3	Minimize window to desktop

```
/*--------------------------------------------------------------------
 * Set minimize button behavior to minimize to desktop
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'HiddenMinWindows'                    /* key name        */
appname   = 'PM_ControlPanel'                     /* application name*/
inifile   = 'USER'                                /* INI file to use */
value     = '3'                                   /* value to insert */
NULL      = '00'x                                 /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'   /* register SysIni */
value = value || NULL                             /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 113. Sample REXX to Set the Minimize Button Behavior

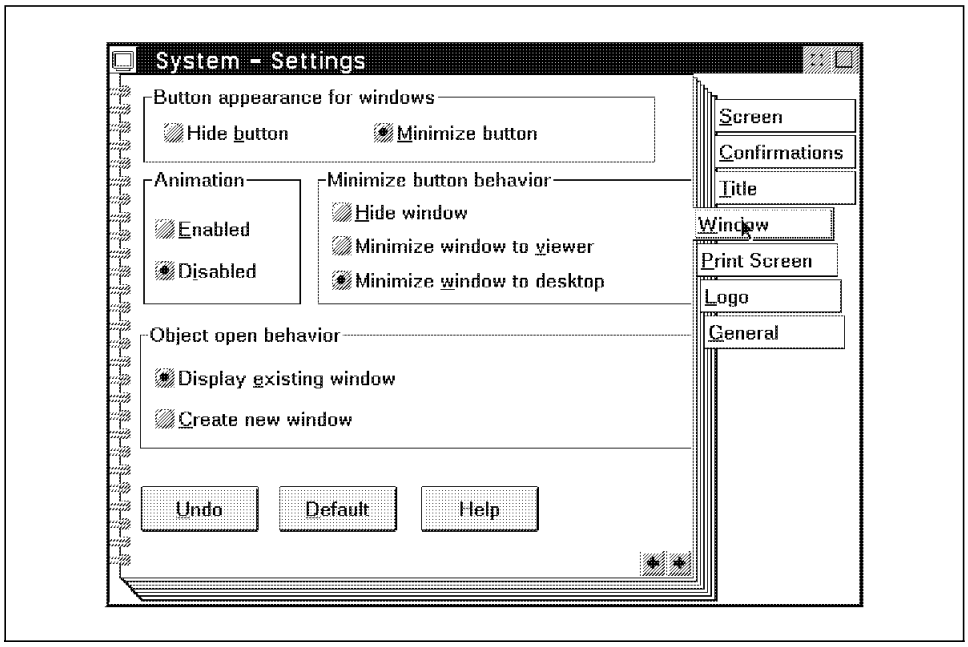


Figure 114. The Window Page from the System Settings

Key Name LogoDisplayTime
Format Integer
Description This value corresponds to the Logo page of the system settings (See Figure 116 on page 141) and controls whether or not applications display their logo windows and if so for how long. The possible values and their meanings are as follows:

Table 31. Logo Display Values

Value	Meaning
-1	Indefinite logo display.
0	No logo display.
number	Time to display the logo in milliseconds.

```
/*--------------------------------------------------------------------
 * Set logo display time to 2 seconds (2000 milliseconds)
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'LogoDisplayTime'                    /* key name        */
appname   = 'PM_ControlPanel'                    /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '2000'                               /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 115. Sample REXX to Set the Logo Display Time

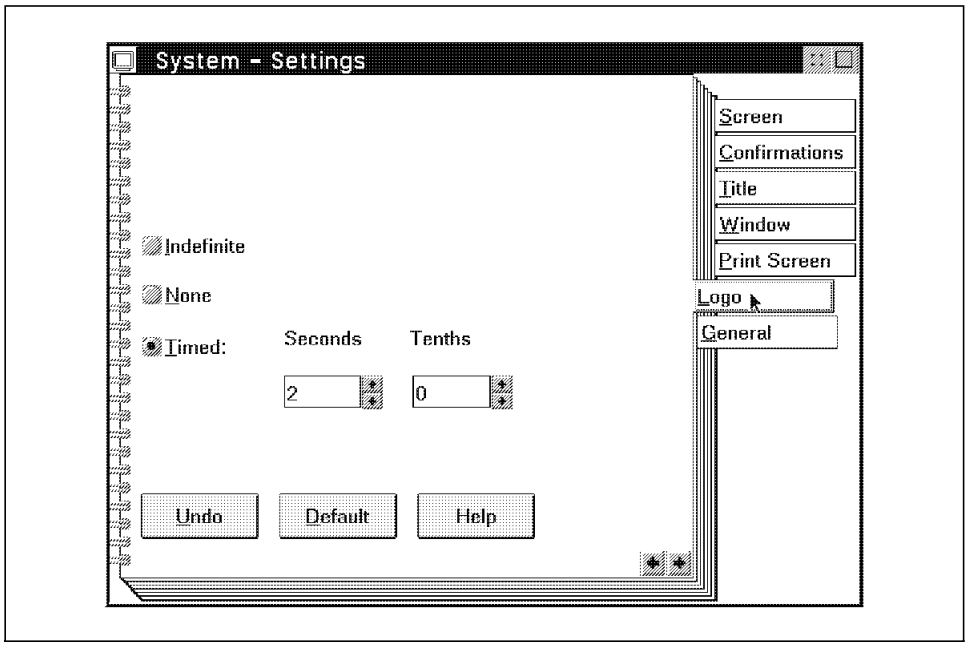


Figure 116. The Logo Page from the System Settings

Key Name MinButtonType
Format Integer
Description This value corresponds to the "Button appearance for windows" radio buttons on the Window page of the System settings (see Figure 114 on page 139) and controls what the minimize buttons look like. I suppose there is an interesting philosophical question in that if a minimize button looks like a hide button, is it still a minimize button? In any case the values are as follows:

Table 32. Minimize Button Values

Value	Corresponding radio button
1	Hide button
2	Minimize button

```
/*--------------------------------------------------------------------
 * Set minimize button appearance to be a hide button
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'MinButtonType'                     /* key name        */
appname   = 'PM_ControlPanel'                   /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '1'                                 /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 117. Sample REXX to Set the Minimize Button Appearance

Key Name NameClash
Format Integer
Description This value corresponds to the "Title clash" radio buttons on the Title page of the System settings (see Figure 119 on page 144) and controls what the system does when you create more than one object in a particular folder with the same type and name. Values are as follows:

Table 33. Name Clash Behavior Values

Value	Corresponding radio button
2	Auto-rename object
8	Replace existing object
16	Prompt for appropriate action

```
/*--------------------------------------------------------------------
 * Set the name clash behavior to replace the existing object
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'NameClash'                          /* key name        */
appname   = 'PM_ControlPanel'                    /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '8'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 118. Sample REXX to Set the Name Clash Behavior

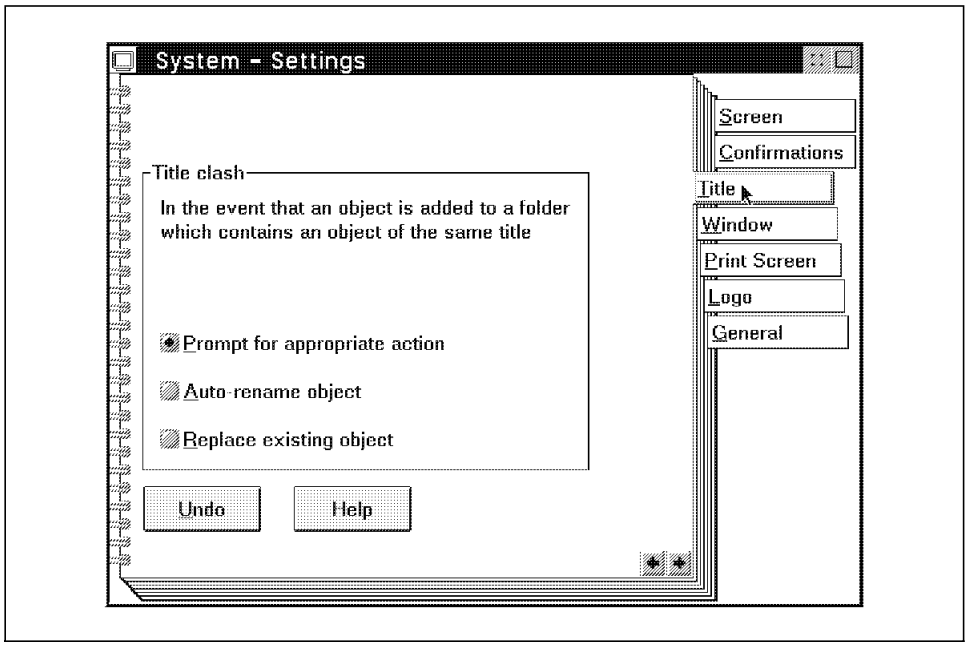


Figure 119. The Title Page from the System Settings

Key Name PrintScreen
Format Integer
Description This value corresponds to the radio buttons on the Print Screen page of the System settings (see Figure 121 on page 146) and controls whether or not the screen can be printed through the use of the Print Screen key. This value is 1 for Enabled and 0 for Disabled. Possible values are as follows:

Table 34. Values for PrintScreen

Value	Description
1	Screen printing is enabled
0	Screen printing is disabled

```
/*--------------------------------------------------------------------
 * Set the Print Screen setting to enabled
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'PrintScreen'                        /* key name        */
appname   = 'PM_ControlPanel'                    /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '1'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 120. Sample REXX to Set the Print Screen Setting

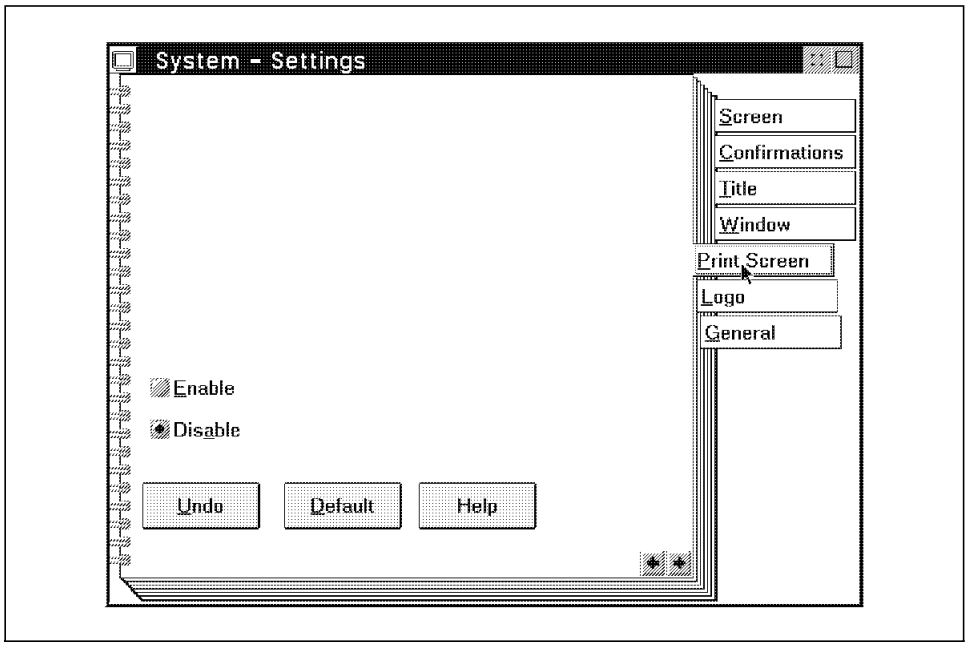


Figure 121. The Window Page from the System Settings

NOTE

This last setting is for the application PM_Workplace and not for PM_ControlPanel.

Key Name CCVIEW
Format Character
Description This setting corresponds to the ″Object open behavior″ radio buttons on the Window page of the System settings (see Figure 114 on page 139). It controls whether or not the default open behavior is to open a new view of an object or to give focus to an already open view. If the value is set to ″ON″, then a new window is created. If the value is set to anything other than ″ON″ or the value is missing, then the existing window is displayed.

```
/*--------------------------------------------------------------------
 * Set default open behavior to create new window
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'CCVIEW'                              /* key name        */
appname   = 'PM_Workplace'                        /* application name*/
inifile   = 'USER'                                /* INI file to use */
value     = 'ON'                                  /* value to insert */
NULL      = '00'x                                 /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'   /* register SysIni */
value = value || NULL                             /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 122. Sample REXX to Set the Default Open Behavior

4.3.2 Changing Country Settings

Country settings are the values in the .INI files that control various national settings, such as the date and currency formats, which tend to vary from country to country. They are usually set at installation time and through the Country object in the Systems Setup folder. If we want to manipulate these values from a program, however, then we can alter the values for the country settings stored in the user .INI file.

All of the country settings are stored in an application called PM_National. In addition to this if you change your country settings after installation time the Workplace Shell will create an application called PM_Default_National which will hold all of the Country settings you specified at installation as a backup.

The different keys you can manipulate and their corresponding Country object equivalents are as follows

Key Name iCountry
Format Integer
Description Country code. This code corresponds to the ″Country″ field on the Country page of the Country settings (see Figure 124 on page 150). Each country listed has a corresponding value (which bears some resemblance to the international telephone country codes) as follows:

Table 35. Country Codes

Country	Code	Country	Code
Arabic Speaking	785	Latin-America	3
Asian English	99	Netherlands	31
Australia	61	Norway	47
Belgium	32	Other Country	0
Brazil	55	Peoples Republic of China	86
Canada (French speaking)	2	Poland	48
Czechoslovakia	42	Portugal	351
Denmark	45	Spain	34
Finland	358	Sweden	46
France	33	Switzerland(FR)	41
Germany	49	Switzerland(GE)	41
Hebrew speaking	972	Taiwan	88
Hungary	36	Turkey	90
Iceland	354	United Kingdom	44
Italy	39	U.S.A.	1
Japan	81	Yugoslavia	38
Korea	82		

```
/*-------------------------------------------------------------------
 * Set the country to Australia ;-)
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iCountry'                          /* key name        */
appname   = 'PM_National'                       /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '61'                                /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 123. Sample REXX to Set the Country to Australia

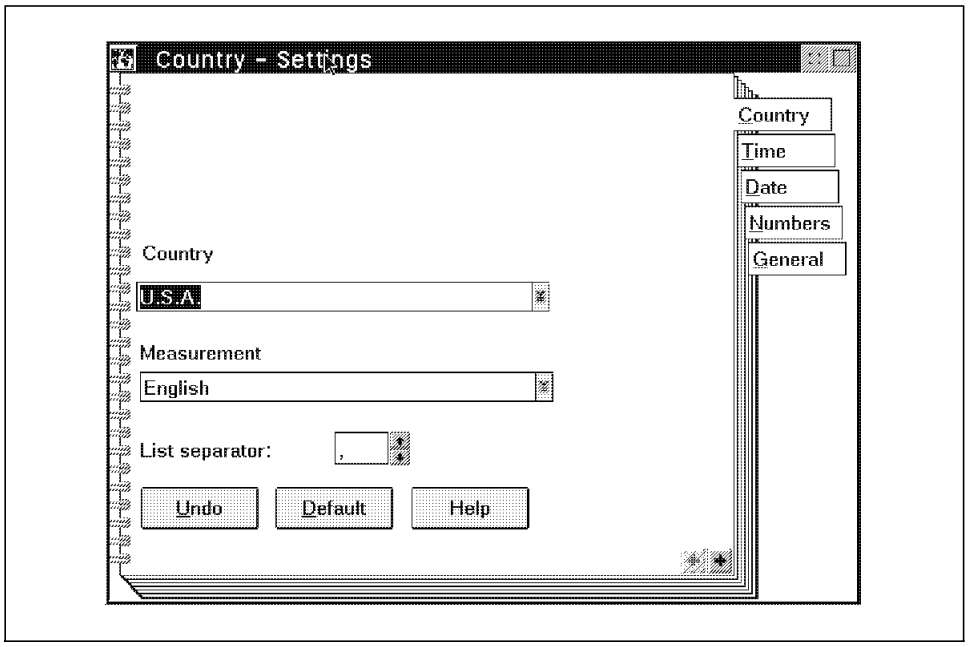


Figure 124. The Country Page from the Country Settings

Key Name iCurrency
Format Integer
Description This value corresponds to the ″Symbol placement″ radio buttons and checkbox on the Numbers page of the Country settings (see Figure 126 on page 151) and controls where the currency symbol is placed in relation to the currency value and whether or not there is an intervening space. The values are as follows:

Table 36. Currency Symbol Placement Codes

Placement	Intervening space	Code
prefix	no	0
suffix	no	1
prefix	yes	2
suffix	yes	3

```
/*--------------------------------------------------------------------
 * Set the currency symbol placement to prefix with no space
 *--------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iCurrency'                          /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '0'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 125. Sample REXX to Set the Currency Symbol Placement

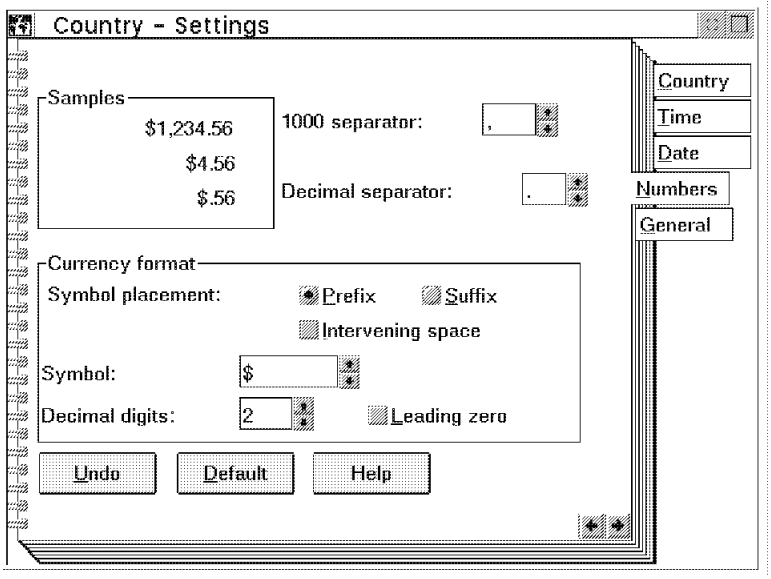


Figure 126. The Numbers Page from the Country Settings

Key Name iDate
Format Integer
Description This setting corresponds to the ″Date order radio buttons″ on the Date page of the Country settings (see Figure 128 on page 153). The values and their meanings are as follows:

Table 37. Date Format Codes

Date Format	Code
Month-Day-Year	0
Day-Month-Year	1
Year-Month-Day	2

```
/*--------------------------------------------------------------------
 * Set the date format to dd-mm-yy
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iDate'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '1'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 127. Sample REXX to Set the Date Format

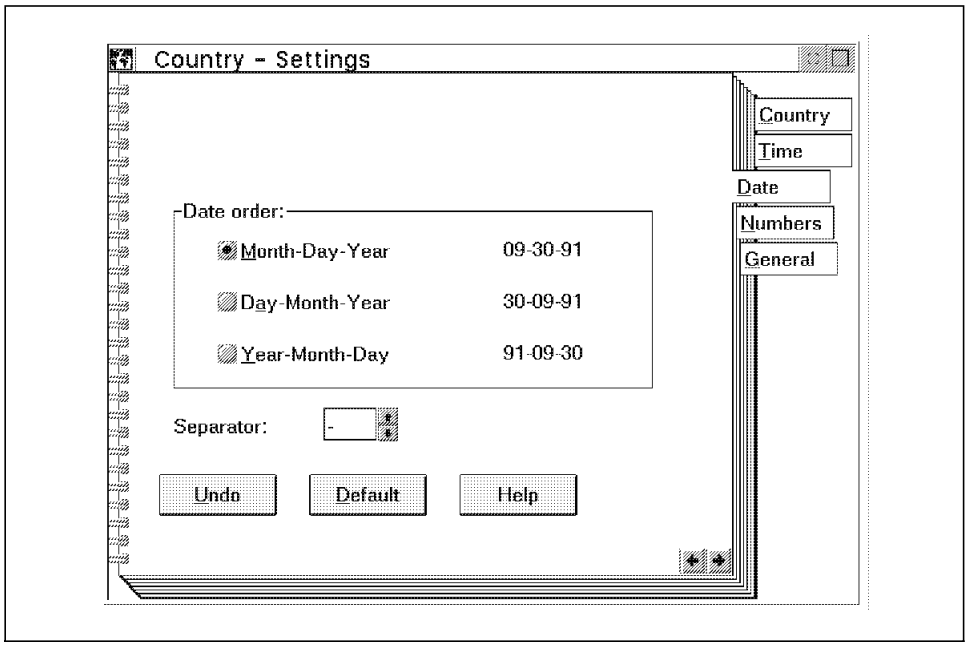


Figure 128. The Date Page from the Country Settings

Key Name iDigits
Format Integer
Description This value corresponds to the "Decimal digits" field in the Currency format section of the Numbers page in the Country settings (see Figure 126 on page 151). The value is the number of decimal digits to use for currency from a minimum of 0 to a maximum of 3.

```
/*--------------------------------------------------------------------
 * Set the number of currency decimal digits to 2
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iDigits'                          /* key name        */
appname   = 'PM_National'                      /* application name*/
inifile   = 'USER'                             /* INI file to use */
value     = '2'                                /* value to insert */
NULL      = '00'x                              /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                          /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 129. Sample REXX to Set the Number of Decimal Currency Digits

Key Name iLzero
Format Integer
Description This value corresponds to the Currency format "Leading zero" check box on the Numbers page of the Country settings (see Figure 126 on page 151). It controls whether a leading zero is included for currency values less than 1. The value is 1 when the check box is checked and 0 when it is unchecked.

```
/*--------------------------------------------------------------------
 * Configure a leading zero for currency values
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iLzero'                              /* key name        */
appname   = 'PM_National'                         /* application name*/
inifile   = 'USER'                                /* INI file to use */
value     = '1'                                   /* value to insert */
NULL      = '00'x                                 /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'   /* register SysIni */
value = value || NULL                             /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 130. Sample REXX to Specify a Leading Zero for Currency Values

Key Name iMeasurement
Format Integer
Description This setting corresponds to the "Measurement" field on the Country page of the Country settings (see Figure 124 on page 150). It controls which measurement system is the default. The values and their meanings are as follows:

Table 38. Measurement System Codes

Measurement System	Code
English	1
Metric	2
Points	3
Picas	4

```
/*--------------------------------------------------------------------
 * Set the measurement system to metric
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iMeasurement'                      /* key name        */
appname   = 'PM_National'                       /* application name*/
inifile   = 'USER'                              /* INI file to use */
value     = '1'                                 /* value to insert */
NULL      = '00'x                               /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                           /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 131. Sample REXX to Set the Measurement System

Key Name iTime
Format Integer
Description This setting corresponds to the "12 hours/24 hours" radio buttons on the Time page of the Country settings (see Figure 133 on page 158). It controls whether a 12 or 24 hour clock is used. The values and their meanings are as follows:

Table 39. Time Format Codes

Time Format	Code
12 hours	0
24 hours	1

```
/*--------------------------------------------------------------------
 * Set the time to be a 12 hour clock
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'iTime'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '0'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 132. Sample REXX to Set a 12 Hour Clock

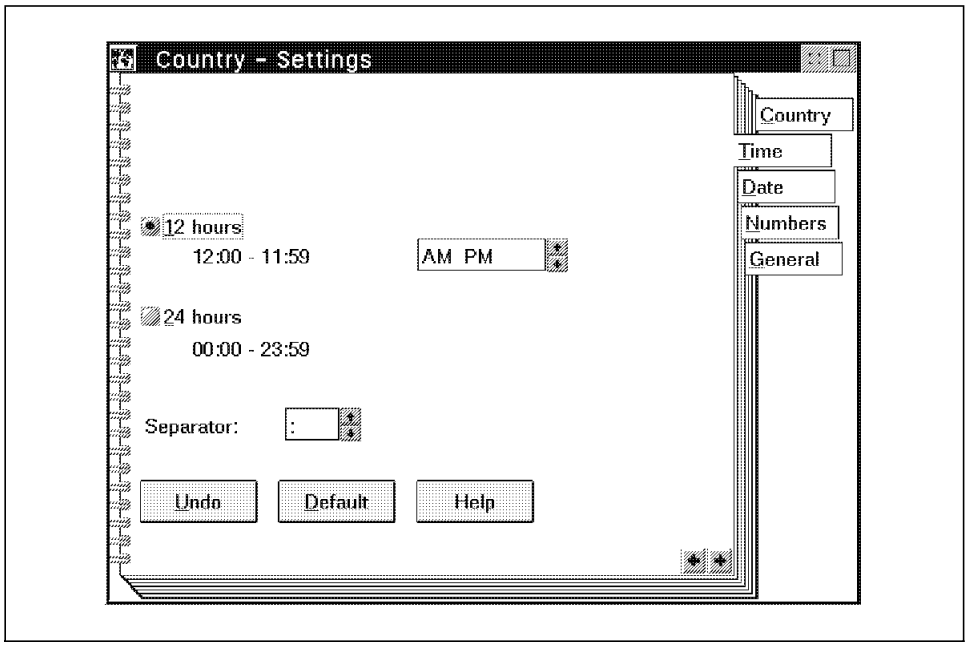


Figure 133. The Time Page from the Country Settings

Key Name sCurrency
Format Character
Description This value corresponds to the "Symbol:" field in the Currency format section of the Numbers page in the Country settings (see Figure 126 on page 151). The value is the string that is used to identify the currency in currency values. This setting can be a maximum of three characters.

```
/*--------------------------------------------------------------------
 * Set the currency symbol to $
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'sCurrency'                          /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '$'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 134. Sample REXX to Set the Currency Symbol

Key Name sDate
Format Character
Description This value corresponds to the "Separator:" field on the Date page of the Country settings (see Figure 126 on page 151). The value is the string that is used to separate the day, month and year of the date. This value can contain a maximum of one character.

```
/*--------------------------------------------------------------------
 * Set the date separator to /
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'sDate'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '/'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 135. Sample REXX to Set the Date Separator

Key Name sDecimal
Format Character
Description This value corresponds to the "Decimal separator:" field on the Numbers page of the Country settings (see Figure 126 on page 151). The value is the string that is used to separate the whole and fractional parts of a number (usually "."). This value can contain a maximum of one character.

```
/*--------------------------------------------------------------------
 * Set the decimal separator to .
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'sDate'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = '/'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 136. Sample REXX to Set the Decimal Separator

Key Name sList
Format Character
Description This value corresponds to the "List separator:" field on the Country page of the Country settings (see Figure 124 on page 150). The character is used to separate items in a series and it can be a maximum of one character.

```
/*--------------------------------------------------------------------
 * Set the list separator to ,
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'sList'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = ','                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 137. Sample REXX to Set the List Separator

Key Name sThousand
Format Character
Description This value corresponds to the "1000 separator:" field on the Numbers page of the Country settings (see Figure 126 on page 151). The value is the string that is placed between every three digits of a long number. This value can contain a maximum of one character.

```
/*--------------------------------------------------------------------
 * Set the thousands separator to ,
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'sThousand'                          /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = ','                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 138. Sample REXX to Set the Thousands Separator

Key Name sTime
Format Character
Description This value corresponds to the "Separator:" field on the Time page of the Country settings (see Figure 133 on page 158). The value is the string that is used to separate the hours, minutes and seconds of the time. This value can contain a maximum of one character.

```
/*--------------------------------------------------------------------
 * Set the time separator to :
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'sTime'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = ':'                                  /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 139. Sample REXX to Set the Time Separator

Key Name s1159
Format Character
Description This setting corresponds to the ″AM PM″ field on the Time page of the Country settings (see Figure 133 on page 158). The value is the string that OS/2 uses with the 12 hour clock to delineate times after midnight and before noon. This value is generally ″am″, ″AM″ or ″ ″ (two spaces) and can be a maximum of three characters.

```
/*--------------------------------------------------------------------
 * Set the midnight to noon indicator to am
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 's1159'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = 'am'                                 /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 140. Sample REXX to Set the Midnight to Noon Indicator

Key Name s2359
Format Character
Description This setting corresponds to the ″AM PM″ field on the Time page of the Country settings (see Figure 133 on page 158). The value is the string that OS/2 uses with the 12 hour clock to delineate times after noon and before midnight. This value is generally ″pm″, ″PM″ or ″ ″ (two spaces) and can be a maximum of three characters.

```
/*--------------------------------------------------------------------
 * Set the noon to midnight indicator to pm
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 's2359'                              /* key name        */
appname   = 'PM_National'                        /* application name*/
inifile   = 'USER'                               /* INI file to use */
value     = 'pm'                                 /* value to insert */
NULL      = '00'x                                /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                            /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 141. Sample REXX to Set the Noon to Midnight Indicator

4.3.3 Changing Your Colors

There are two application names in the .INI files that we need to be concerned about in changing colors. The first one is PM_Colors, which contains all of the current system colors. The second is PM_Default_Colors, which contains all of the default colors. Both of these applications store color data in the same way so a sample program (SETCOLOR.CMD) has been included on the samples diskette distributed with this book to make modifying color information easier.

4.3.3.1 Workplace Shell Color Values

In order to avoid giving each of the more than 65000 colors, available on many systems today, a name like ″A slightly bluer shade of green than the last one but not quite as dark as this other one″, OS/2 uses a coding system.

OS/2 colors can be defined using RGB (Red, Green, Blue) or HSB (Hue, Saturation, Brightness) values. The .RC and .INI files use only the RGB

values to store your color choices. For this reason we will only address RGB values in this section

In an RGB system you give the three colors red, green and blue a value and manipulate the values in different combinations to produce different colors. The OS/2 .INI files store color in 8-bit RGB. That is, each value can range from 0 to 255.

The RGB format used in the .INI files consists of the red value followed by a space, then the green value followed by a space and then the blue value followed by a null character. If any of the values are less than 100 they are padded with spaces to bring them up to three characters so that a set of color values is always a total of 11 characters (not including the terminating null). The following table gives examples of RGB values for the 16 most commonly used colors (the ones found on 16 color systems):

Table 40. Eight Bit RGB Values for Common Colors

Color	Red	Green	Blue	Color Value String
Black	0	0	0	″0 0 0 ″
Dark Blue	0	0	128	″0 0 128″
Dark Green	0	128	0	″0 128 0 ″
Dark Cyan	0	128	128	″0 128 128″
Dark Red	128	0	0	″128 0 0 ″
Dark Pink	128	0	128	″128 0 128″
Brown (Olive)	128	128	0	″128 128 0 ″
Light Gray	204	204	204	″204 204 204″
Dark Gray	128	128	128	″128 128 128″
Blue	0	0	255	″0 0 255″
Green	0	255	0	″0 255 0 ″
Cyan	0	255	255	″0 255 255″
Red	255	0	0	″255 0 0 ″
Pink	255	0	255	″255 0 255″
Yellow	255	255	0	″255 255 0 ″
White	255	255	255	″255 255 255″

The easiest way to determine the RGB values for a desired color is to use the Edit Color dialog of the OS/2 2.x Color Palette located in the System Setup folder (see Figure 142 on page 168) to blend the color directly on the

screen. Then, simply use the values displayed by the Edit Color dialog (see Figure 143 on page 169) once the color is blended to your satisfaction.

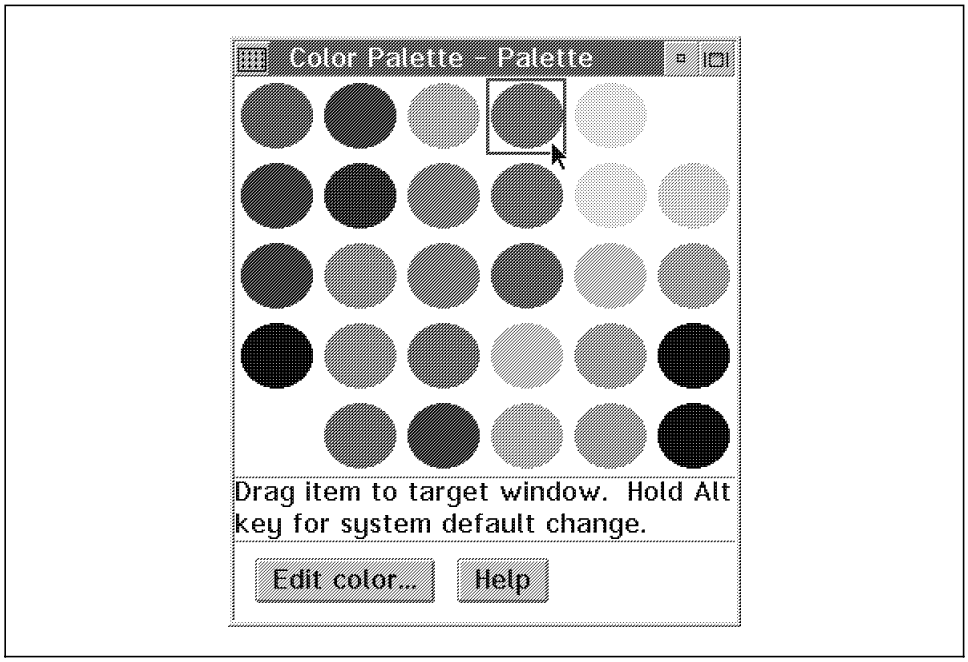


Figure 142. The OS/2 2.x Color Palette

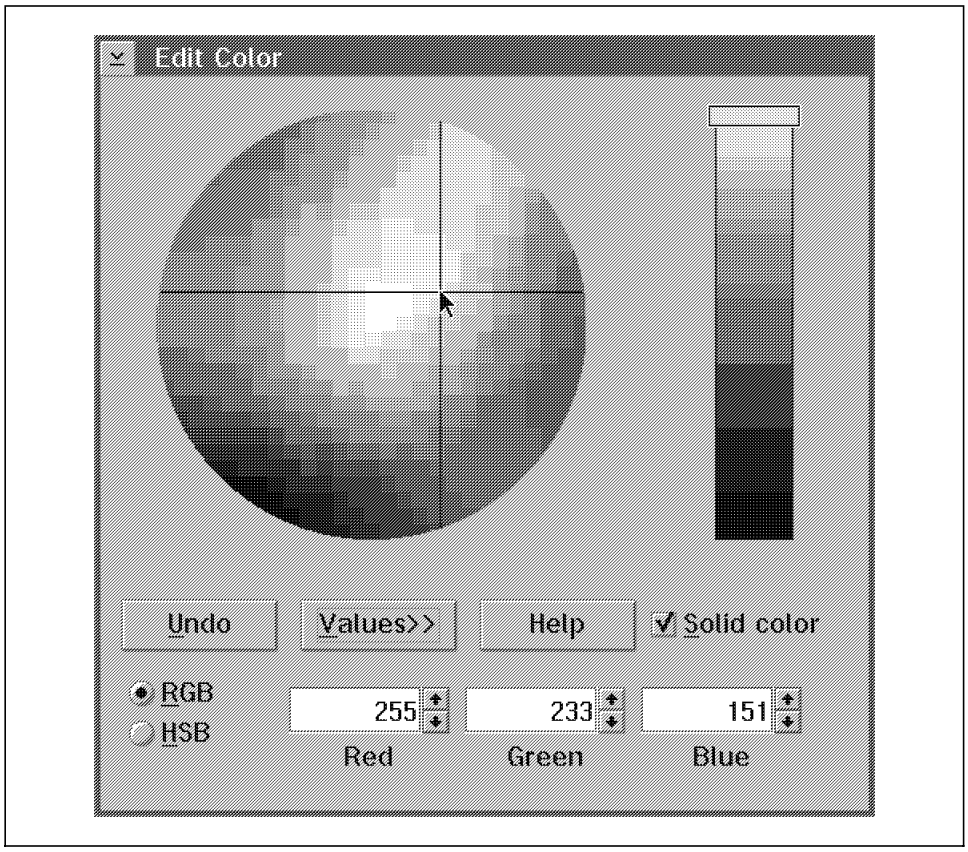


Figure 143. Determining RGB Values for a Displayed Color

4.3.3.2 Changing the System and Default Colors

Both of the applications in OS2.INI (PM_Colors and PM_Default_Colors) contain color information in the RGB format and can be modified using SETCOLOR.CMD (described in 4.3.3.3, "Using SETCOLOR.CMD" on page 170). However, both require slightly different techniques because of how the Workplace Shell treats each application.

The colors you see on your screen are not actually those that are in the .INI file. They are, in fact, a copy of the colors in the .INI that the Workplace Shell makes at boot time. When you change the colors using the Scheme Palettes the Workplace Shell first changes its own copy, so you see the changes immediately, and then writes the changes to the .INI file when you close the Scheme Palette. This means that when you change the colors using the .INI file you must reboot your system in order to see the color changes take

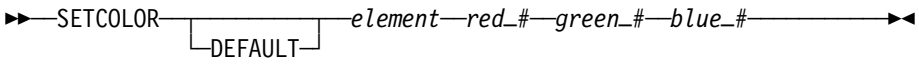
effect. Furthermore, if you close the Scheme Palette after you have modified the .INI file, when you shut down your system all of your .INI file color changes will be lost. This is because the Workplace Shell will have written its version of the .INI over the top of your changes.

On the other hand any changes you make to the default colors will take effect immediately. This is because the Workplace Shell reads the default colors from the INI file each time you click on the Default button in the Edit Scheme dialog.

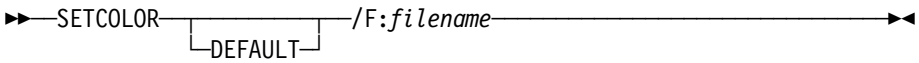
4.3.3.3 Using SETCOLOR.CMD

SETCOLOR.CMD has been written using the same interface to the .INI files that was shown under 4.2, "Reading and Writing Data to the .INI Files" on page 120. It has been included on the sample diskette that accompanies this book along with a number of example files. The program takes either a single set of color values or a file containing many sets of color values and inserts them into the correct places in the user .INI file. The syntax used in calling the program is as follows:

- For a single set of values:



- For sets of values in a file:



where:

DEFAULT Specifies that the default colors are to be modified otherwise the current colors are modified. The DEFAULT keyword is optional.

element Specifies the thing you want to modify the color of. Possible values for element are:

ActiveBorder	HelpText	OutputText
ActiveTitle	HiliteBackground	PageBackground
ActiveTitleText	HiliteForeground	Scrollbar
ActiveTitleTextBgnd	IconText	Shadow
AppWorkspace	InactiveBorder	ShadowHiliteBgnd
ButtonDark	InactiveTitle	ShadowHiliteFgnd
ButtonDefault	InactiveTitleText	ShadowText
ButtonLight	InactiveTitleTextBgnd	TitleBottom
ButtonMiddle	Menu	TitleText
DialogBackground	MenuDisabledText	Window
EntryField	MenuHilite	WindowFrame
FieldBackground	MenuHiliteText	WindowStaticText
HelpBackground	MenuText	WindowText
HelpHilite		

red_# Specifies the red intensity value between 0 and 255

green_# Specifies the green intensity value between 0 and 255

blue_# Specifies the blue intensity value between 0 and 255

filename Specifies the name of the file containing sets of color information. Each line of this file should have the element to modify followed by at least one space followed by the red green and blue intensity values (see Figure 144 on page 172 for an example). Default files for each of the different schemes and the system defaults have been included on the sample diskette.

```
ActiveBorder            255 255 128
ActiveTitle              64 128 128
ActiveTitleText         255 255 255
ActiveTitleTextBgnd      64 128 128
AppWorkspace            255 255 224
Background              204 204 204
ButtonDark              128 128 128
ButtonDefault             0   0   0
ButtonLight             255 255 255
ButtonMiddle            204 204 204
DialogBackground        204 204 204
EntryField              255 255 204
FieldBackground         204 204 204
HelpBackground          255 255 255
HelpHilite                0 128 128
HelpText                  0   0 128
HiliteBackground         96  96  96
HiliteForeground        255 255 255
```

Figure 144. Example of a SETCOLOR Input File

4.3.4 Changing the System Fonts

After changing your colors you might logically want to also change your fonts. Font information is stored in the user .INI file under the application name PM_SystemFonts. Under PM_SystemFonts there can be up to 5 keys:

- DefaultFont
- IconText
- Menus
- WindowText
- WindowTitles

DefaultFont will only be present in the user .INI if you have actually set it. The other 4 keys will be present if you have Alt dragged a font to achieve a global change. Each key stores a font name and point size as its key value. The format is the point size, followed by a period (".") and then the font name with the obligatory terminating null character. For example 8 point Helv would be written "8.Helv". The following table (Table 41 on page 173) lists the default OS/2 system fonts and the corresponding font names used in the .INI file as well as the selectable point sizes.

Table 41 (Page 1 of 2). INI File Font Names for Default System Fonts

Font Name	Selectable Font Palette Point Sizes (Vector Sizes)	Sample .INI File Syntax
Courier	8 10 12 14 18 24 (0-24)	"10.Courier"
Courier Bold	8 10 12 14 18 24 (0-24)	"10.Courier Bold"
Courier Bold Italic	8 10 12 14 18 24 (0-24)	"10.Courier Bold Italic"
Courier Italic	8 10 12 14 18 24 (0-24)	"10.Courier Italic"
Helv	8 10 12 14 18 24 (0-24)	"10.Helv"
Helvetica	8 10 12 14 18 24 (0-24)	"10.Helvetica"
Helvetica Bold	8 10 12 14 18 24 (0-24)	"10.Helvetica Bold"
Helvetica Bold Italic	8 10 12 14 18 24 (0-24)	"10.Helvetica Bold Italic"
Helvetica Italic	8 10 12 14 18 24 (0-24)	"10.Helvetica Italic"
Tms Rmn	8 10 12 14 18 24 (0-24)	"10.Tms Rmn"
Tms Rmn Bold	8 10 12 14 18 24 (0-24)	"10.Tms Rmn Bold"
Tms Rmn Bold Italic	8 10 12 14 18 24 (0-24)	"10.Tms Rmn Bold Italic"
Tms Rmn Italic	8 10 12 14 18 24 (0-24)	"10.Tms Rmn Italic"
Helv	8 10 12 14 18 24 (0-24)	"10.Helv"
Helv Bold	8 10 12 14 18 24 (0-24)	"10.Helv Bold"
Helv Bold Italic	8 10 12 14 18 24 (0-24)	"10.Helv Bold Italic"
Helv Italic	8 10 12 14 18 24 (0-24)	"10.Helv Italic"
Symbol Set	8 10 12 14 18 24 (0-24)	"10.Symbol Set"
System Monospaced	10	"10.System Monospaced"
System Monospaced Bold	10	"10.System Monospaced Bold"
System Monospaced Italic Bold	10	"10.System Monospaced Italic Bold"

Table 41 (Page 2 of 2). INI File Font Names for Default System Fonts

Font Name	Selectable Font Palette Point Sizes (Vector Sizes)	Sample .INI File Syntax
System Monospaced Italic	10	″10.System Monospaced Italic″
System Proportional	10 12	″10.System Proportional″
System Proportional Bold	10 12	″10.System Proportional Bold″
System Proportional Italic Bold	10 12	″10.System Proportional Italic Bold″
System Proportional Italic	10 12	″10.System Proportional Italic″
System Proportional Non-ISO	10 12	″10.System Proportional Non-ISO″
System VIO	2 see 4 5 6 7 8 9 10 11 12 1see 14 15 16 17 18	″10.System VIO″
Times New Roman	8 10 12 14 18 24 (0-24)	″10.Times New Roman″
Times New Roman Bold	8 10 12 14 18 24 (0-24)	″10.Times New Roman Bold″
Times New Roman Bold Italic	8 10 12 14 18 24 (0-24)	″10.Times New Roman Bold Italic″
Times New Roman Italic	8 10 12 14 18 24 (0-24)	″10.Times New Roman Italic″
Tms Rmn	8 10 12 14 18 24 (0-24)	″10.Tms Rmn″

Note:

1. Vector fonts may be given fractional as well as integer point. sizes
2. Vector fonts given a point size of 0 will not be visible.
3. Vector fonts can be given a point size greater than 24 but if these are used on window titles the upper portion of each letter will be cut off to fit it within the title bar.
4. The font names Swiss and Times Roman which are found in the Font Palette are actually Helv and Tms Rmn respectively.

To change the font information stored in the .INI file it is simply a matter of inserting the correct font string into one of the keys. An example of how to do this is given in Figure 145 on page 175. This short program changes the font for some or all (see below for an explanation of which ones) menu items and menu bars. To complete the font change you must reboot your system.

```
/*--------------------------------------------------------------------
 * SETFONT.CMD Set the default font for all system menus to 14 point
 * Times New Roman.
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'Menus'                              /* key name         */
appname   = 'PM_SystemFonts'                     /* application name */
inifile   = 'USER'                               /* INI file to use  */
value     = '14.Times New Roman'                 /* value to insert  */
NULL      = '00'x                                /* null character   */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni  */
value = value || NULL                            /*add null character*/
result = SysIni(inifile, appname, keyname, value)/* set key value    */
if result = 'ERROR:' then
  say 'Error setting .INI file information'
else
  say 'Value for key ' || keyname || ' successfully set to ' || value
exit
```

Figure 145. Program Demonstrating How to Change Font Information

4.3.4.1 Workplace Shell Font Hierarchy

Once you know how to put font information in the .INI file, you need to know what fonts are applied where. The Workplace Shell uses a three tier system of font information to determine what font to use where.

The first layer is on the individual folder/application level. If you drag a font from the font palette to a particular window part that information is either stored by the application or, in the case of a folder or other system object, with that object's other data. If a particular application chooses not to store the font data, the changes you make may be lost when you close the application.

The second layer is the four specific key names in the user .INI file:

- IconText
- Menus
- WindowText
- WindowTitles

Each of these settings controls fonts that have not been set specifically using the method described above. For example, if you set the title bar text on a particular folder to 12 point Courier by dragging a font from the font palette, and then you alter the WindowTitles key value to 8 point Helv and reboot, the folder will still have 12 point Courier on its title bar. The fonts controlled by each setting are fairly self explanatory, IconText controlling icon text, Menus controlling menu bars and menu items, WindowText controlling window text and WindowTitles controlling title bar text.

The third layer is the key value under the DefaultFont key name. If a particular font setting has not been specified individually, and if there is no value under the specific key names relating to the setting, then the font specified under the DefaultFont key name is used.

Warning

A word of warning: when we were testing all of this out we put 8.Helv into the DefaultFont key on our system running at 1024×768. This proved to be very difficult to read and since almost all text on our system was now in this font, the whole system became much more difficult to use. If you change your default font to something like Symbol Set you may have a lot of difficulty recovering!

For a diagram showing how the Workplace Shell determines which font to use, see Figure 146 on page 177.

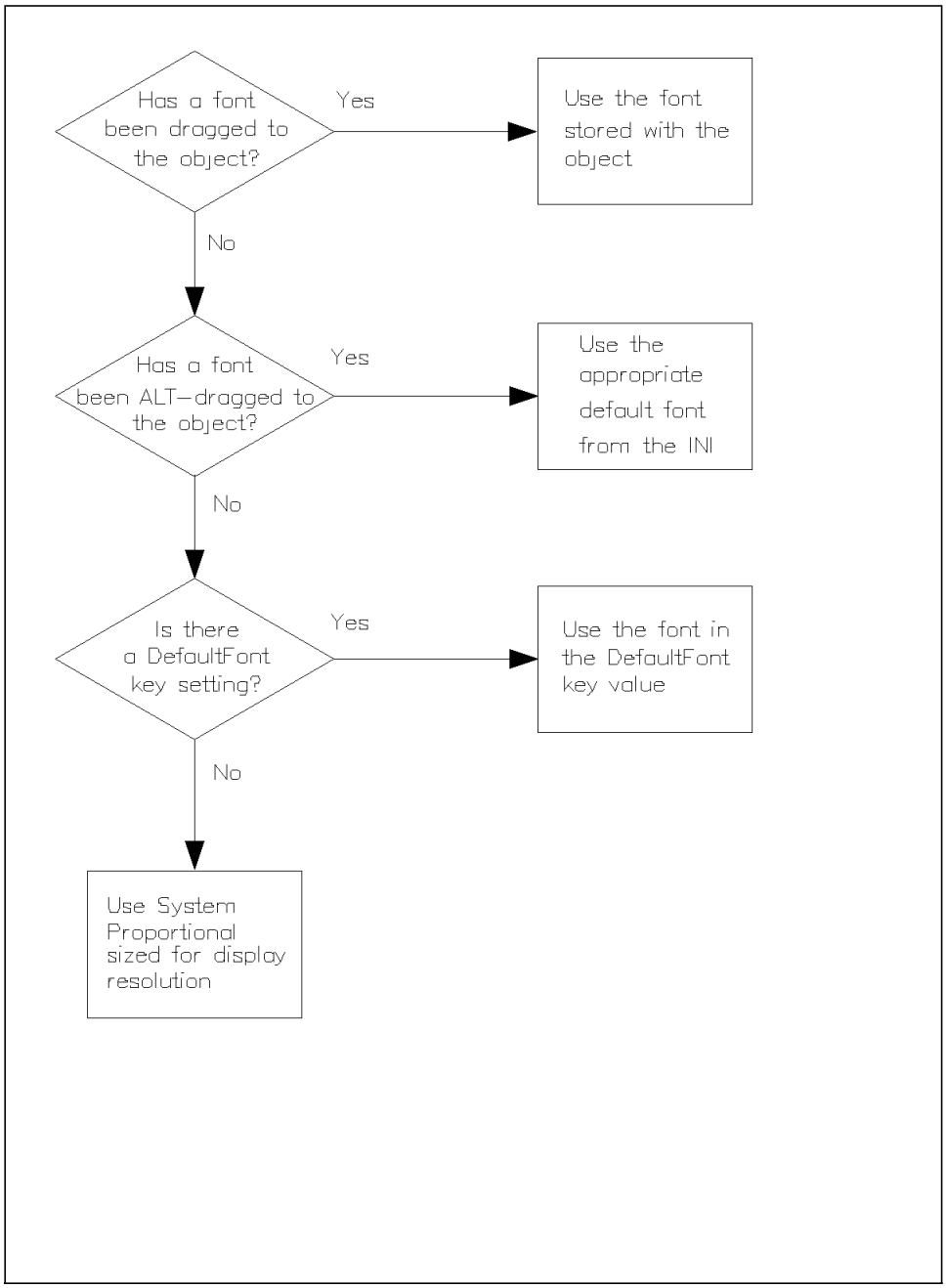


Figure 146. Flowchart Showing How the WPS Chooses Which Font to Use

4.3.5 Changing Your Mouse Mappings

Mouse mappings control which mouse action the system recognizes as the trigger for a certain event. Mouse mappings are all settable from the Mappings page of the Mouse settings in the System Setup Folder (see Figure 147); however, all of the possible settings cannot be achieved from there. There are five mouse mappings that you can set (see Table 42 on page 180).

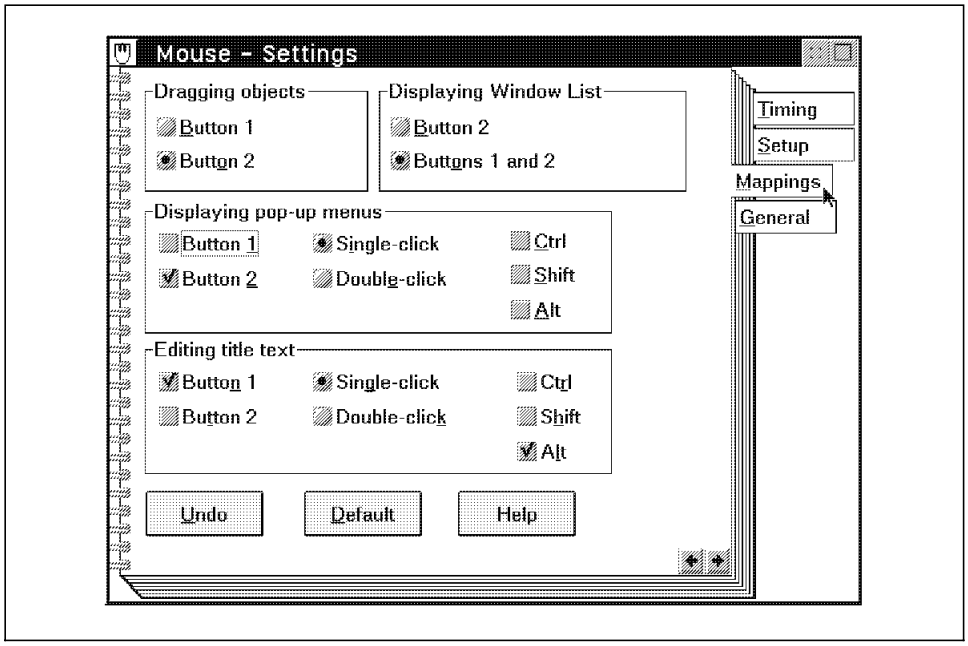


Figure 147. The Mappings Page from the Mouse Settings

Each of these five mappings has a value in the .INI file that is stored under the application PM_ControlPanel. To complicate matters a little each value is made up of 2 subvalues, one for what you are doing with the mouse and one for what you are doing with the keyboard. Also they are stored in binary rather than being stored as a series of characters.

As before we can set these values with a REXX SysIni statement similar to:

```
result = SysIni ([inifile], app, key, val, stem)
```

Figure 148. SysIni Syntax

where inifile is equal to ″USER″, app is equal to ″PM_ControlPanel″, and key is equal to the name of the mapping we wish to set (see Table 42 on page 180).

The ″val″ part of the SysIni call comes in 2 pieces that are concatenated together to give the final value. The first piece tells OS/2 what action you perform with the mouse to trigger the appropriate response (see Table 43 on page 181).

The second piece of the value is the action you perform with the keyboard modifier keys (Shift, Ctrl, and Alt) while you are performing the mouse action (see Table 44 on page 181).

For example, if you wanted to set the action to bring up a context menu to ″Shift and mouse button 1″ you could use the following REXX statements:

```
/* REXX program to set context menu mouse mapping to Shift and mouse
    button 1 */
call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'
call SysIni 'USER','PM_ControlPanel','ContextMenuMouse','13040800'x
exit
```

Figure 149. REXX Program to Set the Context Menu Map to Shift and Mouse Button 1

Note the combination of the click mouse button 1 value (1304) and the Shift key value (0800).

NOTES on Mouse Mappings:

- A value of 0000 for the mouse value will effectively disable that function. For example, you could set the mouse value to 0 for ContextMenuMouse to prevent system users from bringing up context menus.
- TaskListMouseAccess will ignore any keyboard value and bring up the task list based solely on the mouse value regardless of which modifier keys you are pressing.
- The BeginDragMouse and EndDragMouse triggers should be used with paired begin and end mouse drag values; that is, if BeginDragMouse is set to be activated by button 1 then EndDragMouse should be set to be activated by button 1 and likewise for button 2.
- The BeginDragMouse and EndDragMouse triggers should have their keyboard values set to FFFF (ignore modifier keys) as Ctrl, Shift, and Alt drag are all valid drag operations with different meanings to the system. By setting the keyboard value to something other than FFFF you effectively disable one or more of these functions.

C programmers who have access to the OS/2 2.x Toolkit should use the symbolic values instead of the actual numbers to guard against changes. The format of the value for C is a ULONG made up of 2 USHORTs. The upper USHORT contains the keyboard values combined with a logical or while the lower USHORT contains the mouse values. The INP_ values and the WP_ values are defined in PMWIN.H.

Table 42. Mouse Mapping Key Names

BeginDragMouse	the signal that tells the system that you are beginning to drag something with the mouse
ContextMenuMouse	the mouse signal that tells the system you are trying to bring up a context or pop-up menu
EndDragMouse	the signal that tells the system that you have finished dragging something using the mouse
TaskListMouseAccess	the mouse signal that tells the system that you are trying to bring up the task list
TextEditMouse	the mouse signal that tells the system you want to edit an icon's title

Table 43. Mouse Values

Mouse Action	Value in Hex	Corresponding C #define
Start dragging something with mouse button 1	1104	WM_BUTTON1MOTIONSTART
Stop dragging something with mouse button 1	1204	WM_BUTTON1MOTIONEND
Click mouse button 1	1304	WM_BUTTON1CLICK
Double click button 1	7300	WM_BUTTON1DBLCLK
Start dragging something with mouse button 2	1404	WM_BUTTON2MOTIONSTART
Stop dragging something with mouse button 2	1504	WM_BUTTON2MOTIONEND
Click mouse button 2	1604	WM_BUTTON2CLICK
Double click mouse button 2	7600	WM_BUTTON2DBLCLK
Press mouse buttons 1 and 2 at the same time	1004	WM_CHORD

Table 44. Keyboard Values

Keyboard Action	Value in Hex	Corresponding C #define
No modifier key is pressed	0000	INP_NONE
The Shift key is pressed	0800	INP_SHIFT
The Ctrl key is pressed	1000	INP_CTRL
The Alt key is pressed	2000	INP_ALT
The Ctrl and Shift keys are being pressed	1800	n/a
The Ctrl and Alt keys are being pressed	3000	n/a
The Shift and Alt keys are being pressed	2800	n/a
The Ctrl, Alt and Shift keys are being pressed	3800	n/a
The system ignores all modifier keys	FFFF	INP_IGNORE

4.3.6 Changing Other Mouse Settings

The other three mouse settings are much simpler to deal with. Each of them takes an integer value (see Figure 103 on page 126 for an example of how to set a .INI value using SysIni), which is simply a REXX number followed by a terminating null character. The three settings all occur under the application PM_ControlPanel. The key names and functions are as follows:

Key Name DoubleClickSpeed

Description This value corresponds to the "Double-click" setting on the timing page of the Mouse settings (see Figure 151 on page 183). It controls how much time can occur between mouse clicks before the Workplace Shell registers two single clicks instead of one double click. Values can range from 170 for the fastest double click to 1060 for the slowest double click.

```
/*--------------------------------------------------------------------
 * Set the double click speed value to 800
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'DoubleClickSpeed'                  /* key name         */
appname   = 'PM_ControlPanel'                   /* application name */
inifile   = 'USER'                              /* INI file to use  */
value     = '800'                               /* value to insert  */
NULL      = '00'x                               /* null character   */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni  */
value = value || NULL                           /*add null character*/
result = SysIni(inifile, appname, keyname, value)/* set key value    */
exit
```

Figure 150. Sample REXX to Set the Double Click Speed Value to 800

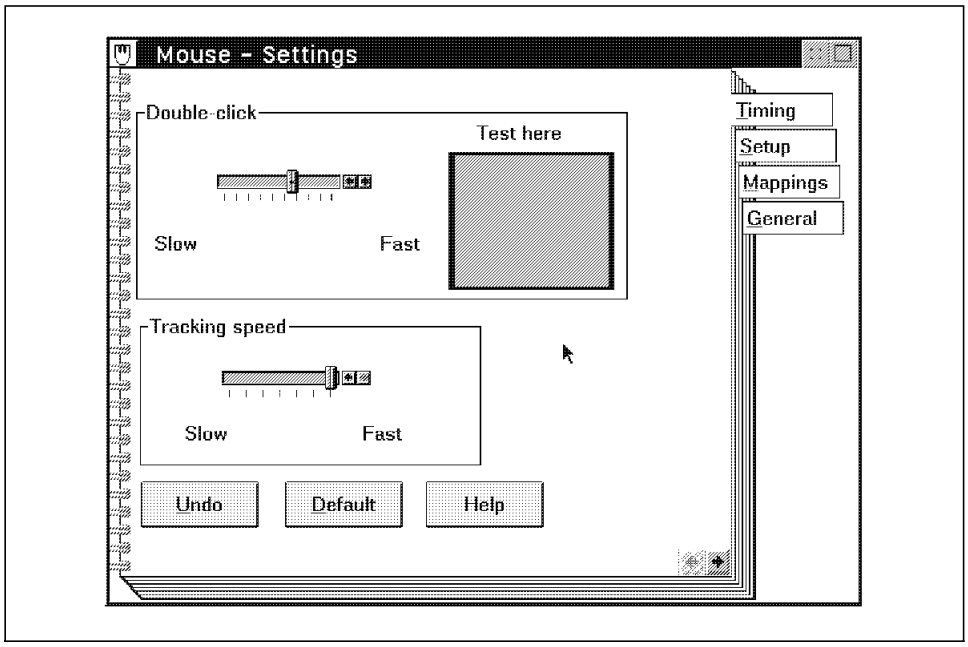


Figure 151. The Timing Page from the Mouse Settings

Key Name MouseTrackingSpeed
Description This value corresponds to the "Tracking speed" setting on the timing page of the Mouse settings (see Figure 151). It controls how far the mouse must move to produce a given movement on the screen. Values can range from 7 for slow tracking to 1 for fast tracking.

```
/*--------------------------------------------------------------------
 * Set the mouse tracking speed setting to 5
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'MouseTrackingSpeed'                /* key name         */
appname   = 'PM_ControlPanel'                   /* application name */
inifile   = 'USER'                              /* INI file to use  */
value     = '5'                                 /* value to insert  */
NULL      = '00'x                               /* null character   */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni  */
value = value || NULL                           /*add null character*/
result = SysIni(inifile, appname, keyname, value)/* set key value    */
exit
```

Figure 152. Sample REXX to Set the Mouse Tracking Speed to 5

Key Name SwapMouseButtons

Description This value corresponds to the setting on the Setup page of the Mouse settings (see Figure 154 on page 185). It controls which button is considered button 1 and which button is considered button 2. The data value is 1 for a left handed mouse and 0 for a right handed mouse.

```
/*--------------------------------------------------------------------
 * Set the mouse up as a left handed mouse
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'SwapMouseButtons'                  /* key name         */
appname   = 'PM_ControlPanel'                   /* application name */
inifile   = 'USER'                              /* INI file to use  */
value     = '1'                                 /* value to insert  */
NULL      = '00'x                               /* null character   */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni' /* register SysIni  */
value = value || NULL                           /*add null character*/
result = SysIni(inifile, appname, keyname, value)/* set key value    */
exit
```

Figure 153. Sample REXX to Set the Mouse Orientation

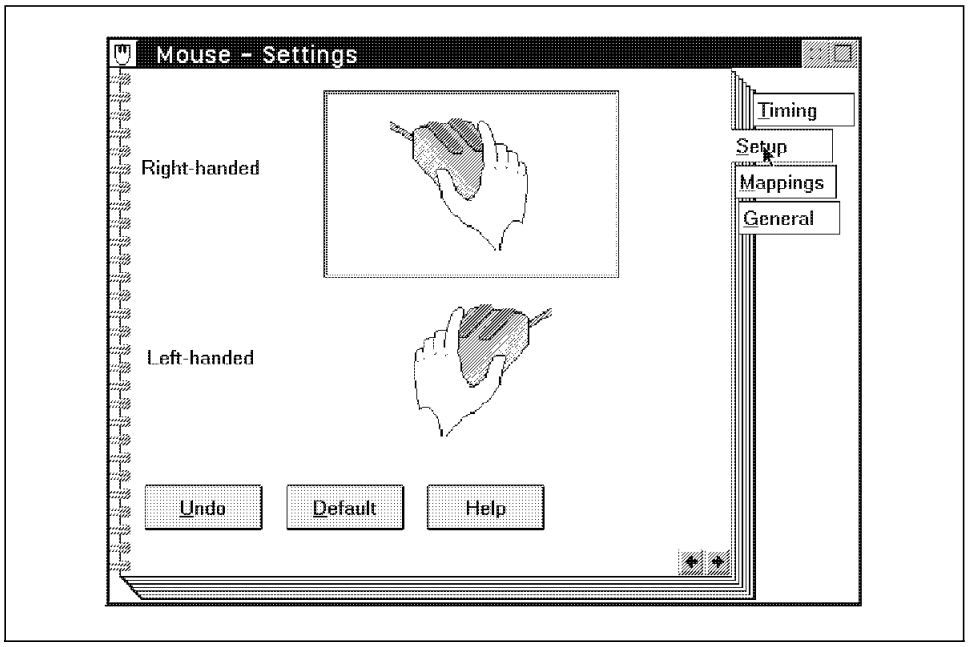


Figure 154. The Setup Page from the Mouse Settings

4.3.7 Changing Your Keyboard Mappings

Keyboard mappings are similar to mouse mappings in that they tell the system what combination of keys, signals a particular action. There are two keyboard mappings that you can set and they are both settable from the Mappings page of the Keyboard settings (see Figure 155 on page 186) in the System Setup Folder. Their names and definitions are included in Table 45 on page 187.

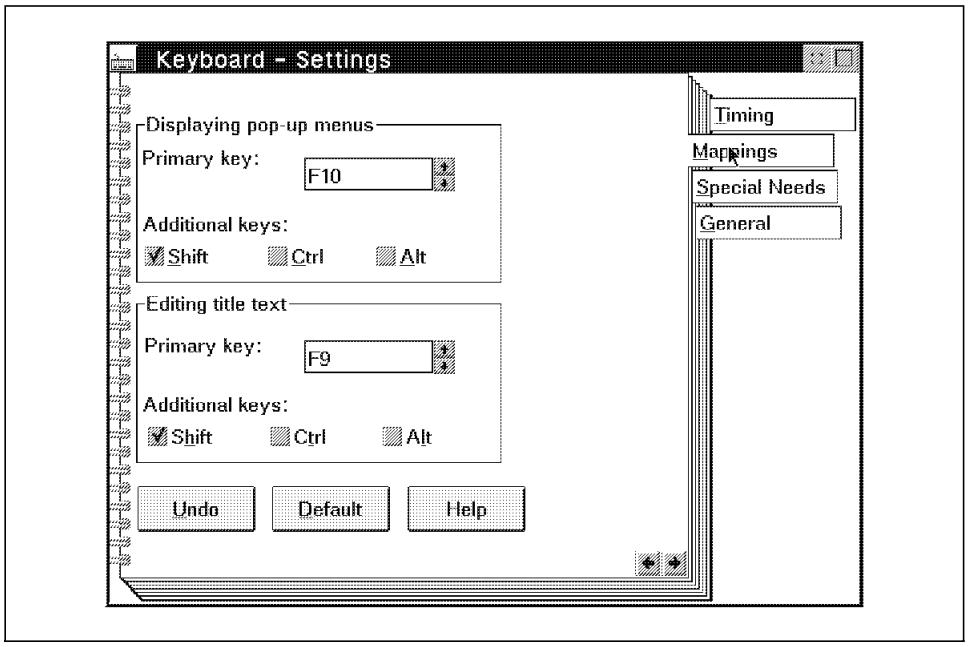


Figure 155. The Mappings Page from the Keyboard Settings

Both of these mappings have a value in the .INI file that is stored under the application PM_ControlPanel. As with the mouse mapping settings each value in the .INI file is a binary value made up of 2 sub parts, one for the key pressed and one for the presence of modifier keys (Ctrl, Alt, and Shift).

As before we can set these values with a REXX SysIni statement similar to:

```
result = SysIni ([inifile], app, key, val, stem)
```

Figure 156. SysIni Syntax

where inifile is equal to ″USER″, app is equal to ″PM_ControlPanel″ and key is equal to the name of the mapping we wish to set (see Table 45 on page 187).

The val part of the SysIni call comes in two pieces that are concatenated together to give the final value. The first piece tells OS/2 which key you press to trigger the appropriate response (see Table 46 on page 188).

The second piece of the value is the action you perform with the keyboard modifier keys (Shift, Ctrl, and Alt) while you are pressing the key (see Table 47 on page 189).

For example, if you wanted to set the action to bring up a context menu to Shift-Page Up you could use the following REXX statements:

```
/* REXX program to set the context menu keyboard mapping to shift and
 * the Page Up key                                                    */
call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'
call SysIni 'USER','PM_ControlPanel','ContextMenuKB','11000A01'x
exit
```

Figure 157. REXX to Set the Context Menu Keyboard Mapping to Shift-Page Up

Note the combination of the Page up key value (1100) and the Shift key value (0A01).

C programmers who have access to the OS/2 2.x Toolkit should use the symbolic values instead of the actual numbers to guard against changes. The format of the value for C is a ULONG made up of 2 USHORTs. The upper USHORT contains the keyboard modifier flags which are the same as those described in *The OS/2 2.0 Technical Library Presentation Manager Programming Reference* Volume III under the fsflags field on the WM_CHAR message. The values for the KC_SHIFT, KC_CTRL and KC_ALT flags, if used, are combined with the KC_VIRTUALKEY and KC_LONEKEY values using logical or. The lower USHORT contains the virtual key values. The VK_ values and the KC_ values are defined in PMWIN.H, which is part of the OS/2 2.x Toolkit.

Table 45. Keyboard Mapping Key Names

ContextMenuKB	the keyboard signal that tells the system you are trying to bring up a context or pop-up menu.
TextEditKB	the keyboard signal that tells the system you want to edit an icon's title.

Table 46. Keyboard Values

Key	Value in Hex	Corresponding C #define
Backspace	0500	VK_BACKSPACE
Tab	0600	VK_TAB
Pause	0D00	VK_PAUSE
Caps Lock	0E00	VK_CAPSLOCK
Space Bar	1000	VK_SPACE
Page Up	1100	VK_PAGEUP
Page Down	1200	VK_PAGEDOWN
End	1300	VK_END
Home	1400	VK_HOME
left arrow	1500	VK_LEFT
up arrow	1600	VK_UP
right arrow	1700	VK_RIGHT
down arrow	1800	VK_DOWN
Print Screen	1900	VK_PRINTSCRN
Insert	1A00	VK_INSERT
Delete	1B00	VK_DELETE
Scroll Lock	1C00	VK_SCRLLOCK
Num Lock	1D00	VK_NUMLOCK
Enter	1E00	VK_ENTER
F1	2000	VK_F1
F2	2100	VK_F2
F3	2200	VK_F3
F4	2300	VK_F4
F5	2400	VK_F5
F6	2500	VK_F6
F7	2600	VK_F7
F8	2700	VK_F8
F9	2800	VK_F9
F10	2900	VK_F10
F11	2A00	VK_F11
F12	2B00	VK_F12

Table 47. Keyboard Modifier Values

Keyboard Action	Value in Hexadecimal	Corresponding C #define
No modifier key is pressed	0201	n/a
The Shift key is pressed	0A01	KC_SHIFT
The Ctrl key is pressed	1201	KC_CTRL
The Alt key is pressed	2201	KC_ALT
The Ctrl and Shift keys are being pressed	1A01	n/a
The Ctrl and Alt keys are being pressed	3201	n/a
The Shift and Alt keys are being pressed	2A01	n/a
The Ctrl, Alt and Shift keys are being pressed	3A01	n/a

4.3.8 Workplace Shell Setup for Users with Special Needs

The Workplace Shell provides a number of settings to allow special needs users to interact with the shell more easily. These settings are usually set using the Special Needs page of the Keyboard settings (see Figure 158 on page 190), but they can also be set by placing data in the user INI file.

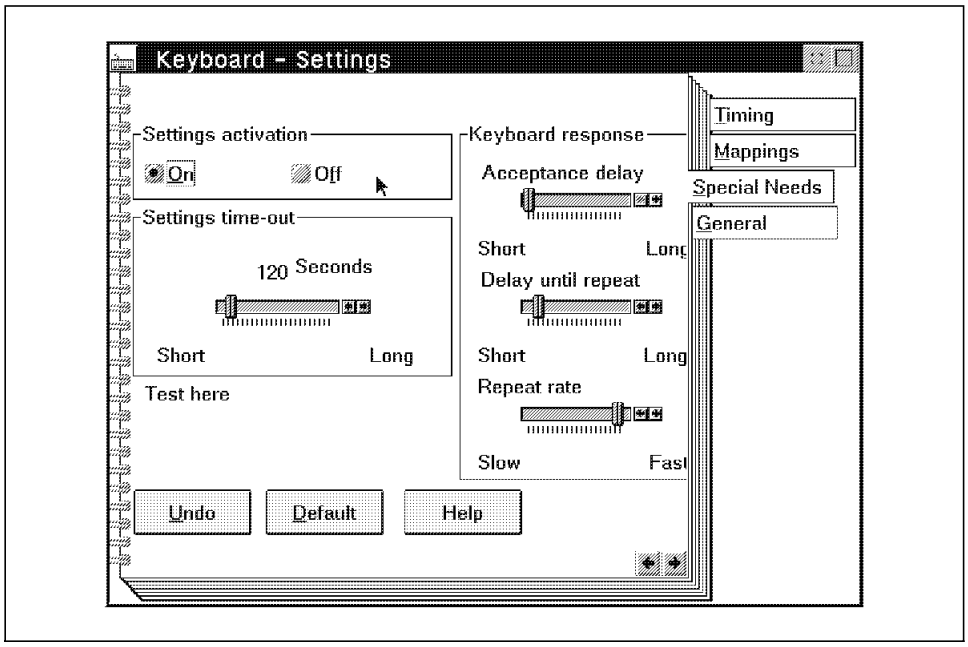


Figure 158. The Special Needs Page from the Keyboard Settings

All of the .INI file information is stored under the application name ″PM_AlternateInputMethods″ in binary form. With the exception of the activation setting ,AIM_Active, each setting is a binary number occupying 2 bytes. C programmers will be able to use the data type USHORT to access these settings whereas REXX users should use the code fragments presented below which reformat REXX numbers into the correct binary form before inserting them into the .INI file.

Key Name AIM_Active
Format Binary
Description This value corresponds to the "Settings Activation" radio buttons on the Special Needs page of the Keyboard settings (see Figure 158 on page 190). If this value is set to TRUE (see Table 48) then the other special needs settings take effect, otherwise they have no effect. The following table lists the permitted values:

Table 48. Special Needs Values

Value	Meaning
0100	Special Needs settings are in effect
0000	Special Needs settings have no effect

```
/*--------------------------------------------------------------------
 * Activate the special needs settings
 *-------------------------------------------------------------------*/
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
result = SysIni('USER','PM_AlternateInputMethods','AIM_Active','0100')
                                                  /* insert new value*/
exit
```

Figure 159. Sample REXX to Activate the Special Needs Settings

Key Name AIM_FKAccept
Format Binary
Description This value corresponds to the Keyboard response "Acceptance delay" setting on the Special Needs page of the Keyboard Settings (see Figure 158 on page 190). It controls how long a key on the keyboard must be held down before it is accepted. Each tick on the slider corresponds to an increment of 250 up to a maximum of 10000 indicating the delay in milliseconds (that is, 0 to 10 seconds).

```
/*--------------------------------------------------------------------
 * Set Keyboard Response Acceptance Delay to 1500
 *-------------------------------------------------------------------*/
Delay = 1500                                    /* Delay Value     */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* Register SysIni */
Delay = d2x(Delay,4)                            /* Convert to hex  */
Delay = right(Delay,2) || left(Delay,2)         /* reverse bytes   */
Delay = x2c(Delay)                              /* convert to char */
result = SysIni('USER','PM_AlternateInputMethods','AIM_FKAccept',Delay)
                                                /* insert new value*/
exit
```

Figure 160. Sample REXX to Set Keyboard Response Acceptance Delay to 1500

Key Name AIM_FKDelay
Format Binary
Description This value corresponds to the Keyboard response ″Delay until repeat″ setting on the Special Needs page of the Keyboard Settings (see Figure 158 on page 190). It controls how long a key must be held down until the Workplace Shell begins to repeat it. Each tick on the slider corresponds to an increment of 250 up to a maximum of 10000 indicating the delay in milliseconds (that is, 0 to 10 seconds).

```
/*--------------------------------------------------------------------
 * Set keyboard response delay until repeat to 1993
 *-------------------------------------------------------------------*/
Delay = 1993                                    /* Delay Value     */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* Register SysIni */
Delay = d2x(Delay,4)                            /* Convert to hex  */
Delay = right(Delay,2) || left(Delay,2)         /* reverse bytes   */
Delay = x2c(Delay)                              /* convert to char */
result = SysIni('USER','PM_AlternateInputMethods','AIM_FKDelay',Delay)
                                                /* insert new value*/
exit
```

Figure 161. Sample REXX to Set Keyboard Response Delay

Key Name AIM_FKRate
Format Binary
Description This value corresponds to the Keyboard response "Repeat rate" setting on the Special Needs page of the Keyboard Settings (see Figure 158 on page 190). Each tick on the slider corresponds to an increment of 250 milliseconds up to a maximum of 10000 milliseconds (that is, 10 seconds) with a minimum of 0. The time is the number of milliseconds after which the key will repeat with 0 representing normal repeat rate. For example, if you specified a value of 3000 then a key that was held down would repeat after every 3 seconds.

```
/*--------------------------------------------------------------------
 * Set keyboard repeat rate to 3000
 *-------------------------------------------------------------------*/
Rate = 3000                                     /* Rate Value      */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* Register SysIni */
Rate = d2x(Rate,4)                              /* Convert to hex  */
Rate = right(Rate,2) || left(Rate,2)            /* reverse bytes   */
Rate = x2c(Rate)                                /* convert to char */
result = SysIni('USER','PM_AlternateInputMethods','AIM_FKRate',Rate)
                                                /* insert new value*/
exit
```

Figure 162. Sample REXX to Set Keyboard Repeat Rate to 3 Seconds

Key Name AIM_TimeOut
Format Binary
Description This value corresponds to the "Settings time-out" field on the Special Needs page of the Keyboard settings (see Figure 158 on page 190). The value is the number of seconds of inactivity that must elapse before the special needs settings are inactivated and normal keyboard function resumes. If a value of 0 is entered, then the settings do not time out and must be turned off manually. Maximum value is 1780 (around 30 minutes until timeout).

```
/*--------------------------------------------------------------------
 * Set special needs timeout to 600
 *-------------------------------------------------------------------*/
Rate = 3000                                     /* Rate Value      */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* Register SysIni */
Rate = d2x(Rate,4)                              /* Convert to hex  */
Rate = right(Rate,2) || left(Rate,2)            /* reverse bytes   */
Rate = x2c(Rate)                                /* convert to char */
result = SysIni('USER','PM_AlternateInputMethods','AIM_FKRate',Rate)
                                                /* insert new value*/
exit
```

Figure 163. Sample REXX to Set Special Needs Timeout to 10 Minutes

4.3.9 Installing Your Own Fonts

If you have a need to install your own fonts without using the standard dialogs, then this section is for you. The Workplace Shell stores font information in an application called PM_Fonts. Each key name under this application is the name of a font file that the Workplace Shell can use, and the key values are paths to that file.

For ATM fonts the Workplace Shell needs to generate a .OFM file at install time. These fonts cannot be installed correctly using this method unless you have already installed the font on another system and generated the .OFM file.

To install your own fonts perform the following steps:

1. Copy the font files to an appropriate directory. If you have bitmapped fonts, this will probably consist of one or more .FON files. If you have ATM fonts, this will probably consist of a .OFM file and a .PFB file for each font (the .OFM file having been generated on another OS/2 system). There may also be a .PFM file but these files are only used by WIN-OS2.
2. For each font, update PM_Fonts by creating a new key name with the following form:
   - For bitmapped fonts the key name is the name of the .FON file minus the extension. For example, if you had a .FON file called FOO.FON then you would create a key called FOO.
   - For ATM fonts the key name is the name of the .OFM file.

   In each case the key value that should be used is the drive and path of the font file followed by the obligatory terminating null character (hex 00). To accomplish this for the vector font BAR with a font file BAR.OFM located in the directory C:MYFONTS you could use the following REXX:

```
/*-------------------------------------------------------------------
 * Install the font BAR in C:\MYFONTS
 *-------------------------------------------------------------------*/
/* initialize variables */
keyname   = 'BAR.OFM'                          /* key name        */
appname   = 'PM_Fonts'                         /* application name*/
inifile   = 'USER'                             /* INI file to use */
value     = 'C:\MYFONTS\BAR.OFM'   /* value to insert */
NULL      = '00'x                              /* null character  */
call   RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* register SysIni */
value = value || NULL                          /* add null        */
result = SysIni(inifile, appname, keyname, value)/* set key value   */
exit
```

Figure 164. Sample REXX to Install the Font BAR in C:MYFONTS

4.4 Binary Data in the .INI Files

Note

This section is only for those people who would like to understand the intricacies of how binary data is stored in the .INI files. All the other sections include REXX code which takes this information into account and provide tables of pre-formatted data where necessary.

Most of the data types in the .INI files have been relatively easy to deal with, requiring only that we add a terminating null character to our data. However, some are not as easy to handle and a brief discussion of how the C language stores data is in order.

When C stores numbers it stores them in binary format; that is, it stores numbers as binary values as opposed to the binary codes that represent the numeral characters. For example, the binary code for the number 205 is 11001101, whereas the binary codes for the three numeral characters are 00110010, 00110000, and 00110101. As you can see, the binary representation takes up a lot less space. These numbers are usually expressed in hexadecimal so 205 becomes CD and the numeral character codes become 32, 30, and 35, respectively. REXX provides the conversion function D2X to enable us to convert decimal numbers to hexadecimal.

If life was as simple as this it wouldn't be too bad but there are a couple of things we must also consider. In binary format we need to know the length of the number we are trying to store so we know how many hexadecimal digits to write. Each byte in the .INI file can be represented by 8 binary digits or 2 hexadecimal digits so if we know how long a number should be in bytes we simply multiply by 2 to get the number of hexadecimal digits to write. Apart from character data, the three data types most commonly used in the .INI files are called USHORT (unsigned short), ULONG (unsigned long) and BOOL (boolean). Table 49 shows the respective lengths of each of these data types.

Table 49. Lengths of Various Data Types

Data Type Name	**Data Type Length in Bytes**
USHORT	2
ULONG	4
BOOL	4

The second complication is that OS/2 stores each of these bytes backwards, yes, in reverse order. So going back to our earlier example of storing the number 205, we first convert it to hexadecimal so it becomes CD. If we then want to store it in a USHORT we need to add leading zeros to make it up to 4 hexadecimal digits (a USHORT is 2 bytes in length and we multiply by 2) so it becomes 00CD. Finally we need to reverse the byte order so 205 will be stored as CD00 (note that the values within the individual bytes are not reversed, just the bytes themselves).

In a REXX program we could write this as ″CD00″x or we could leave it as the characters ″CD00″ and then use the function X2C to convert it to binary before we write it to the .INI file. Generally we store numbers in the hexadecimal character form initially so that it is easier to reverse the byte order and then convert them to their final binary form.

The REXX function D2X, as well as providing decimal to hexadecimal character format conversion, allows us to specify the number of hexadecimal digits to use so the only part of the process not covered is the byte reversal. The REXX function reverse shown in Figure 165 on page 199 performs this function by taking a number in hexadecimal character format and reversing the byte order. The number would then have to be converted using X2C before insertion in a .INI file.

```
reverse: procedure
/*--------------------------------------------------------------------
 * reverse: REXX function to reverse the byte order in a hexadecimal
 *          string
 *
 * input  : a hexadecimal string with no blanks
 * output : a hexadecimal string giving the bytes in the input in
 *          reverse order. If the input string contains an odd number
 *          of digits a leading zero is inserted before conversion.
 *          If the input string is not a valid hexadecimal string
 *          then a null string is returned.
 *--------------------------------------------------------------------*/
arg input_string .

result = ''

if datatype(input_string, 'X') = 1 then /* we have a valid hex string */
  do
    len = length(input_string)
    if (len // 2) <> 0 then /* odd number of digits */
      do
        input_string = '0' || input_string
        len = len + 1
      end /* if */
    if len < 4 then
      do
        result = input_string
      end /* if */
    else
      do i = 1 to len - 1 by 2
        result = substr(input_string, i, 2) || result
      end /* else */
  end /* if */

return result
```

Figure 165. Reverse Function for Reversing Byte Order

Chapter 5. CID and The WorkPlace Shell

With the increased interest in being able to install and support workstations from a central location via a LAN we decided to investigate a way to customize a desktop during the CID installation. We tested three different ways to do this:

- Customizing the CONFIG.SYS
- Using customized .RC files
- Customizing the .INI files

This section is not going to go through the basics of doing a CID installation; there are various manuals that cover the actual CID process. A good basic knowledge of CID is assumed for this section. If you find that you require additional information on the CID process please reference the list of related publications at the front of this book.

5.1 CONFIG.SYS

If you decide to use the CONFIG.SYS to customize the desktop of your target systems, you should first read Chapter 2, “Changing the Desktop via the CONFIG.SYS” on page 5. This will give you an understanding of the changes you can make to your CONFIG.SYS, which will customize your desktop. You will find that the CONFIG.SYS method of customizing your desk top will give you the opportunity to change the large overall look of the desktop but does not allow you to make minor changes like colors, screen positions, etc. For example, if you change your SET RUNWORKPLACE= statement so that an application becomes your shell, you will be able to limit what is actually on your desktop but not where it is positioned on the desktop or what color the desktop is. The PROTSHELL= statement can also be used for this, as per Chapter 2, “Changing the Desktop via the CONFIG.SYS” on page 5.

We tested two ways of getting a customized CONFIG.SYS up and running on your target systems during the CID installation process:

- The ConfigSysLine response file keyword
- The UserExit response file keyword

5.1.1 ConfigSysLine

The ConfigSysLine keyword allows you to add a line to your CONFIG.SYS. This means that if you use this keyword to add the SET RUNWORKPLACE=E:\DESCRIBE\DESCRIBE.EXE″, to the CONFIG.SYS on the target system, the target system will come up with Describe running after you reboot the machine from the CID installation. Also, if you check the CONFIG.SYS on the system you will find that it contains two SET RUNWORKPLACE= statements, the original and the one that was added by the CID install. The second line, which is pointing to what ever you set it for, will override the original line which points to the PMSHELL.EXE.

If you want to have more than one application running you can either have a CMD file in the Set RUNWORKPLACE= or use the UserExit keyword to copy a ready made CMD file to your system.

The PROTSHELL= statement does not work as well with this procedure. This is because this procedure leaves the SET RUNWORKPLACE=C:\OS2\PMSHELL.EXE statement in the CONFIG.SYS and it will bring up the default PM objects in addition to the application that the PROTSHELL= statement starts. The way to get around this is to put a second ConfigSysLine keyword in your response file. This ConfigSysLine should be **ConfigSysLine=SET RUNWORKPLCE=** . This will override the original SET RUNWORKPLACE statement and because the second statement is blank it will not start anything extra.

5.1.2 UserExit and CMD File

This method uses the UserExit keyword in the response file to call a CMD file. The CMD file could look similar to Figure 166.

```
COPY X:\CONFIG.1 C:\CONFIG.SYS
COPY X:\STARTUP.1 C:\STARTUP.CMD
```

Figure 166. Sample CONFIG.SYS CMD File

The CMD file will copy a previously customized CONFIG.SYS from a directory on the CID server to the boot drive of the target system. This will replace the original CONFIG.SYS file and this way when you boot the system it will use the customized CONFIG.SYS and your desktop will be set up. The same CMD file can be used to copy a STARTUP.CMD file to the target system thereby allowing any additional applications to be started when the system is rebooted.

5.2 RC Files

The RC files are covered in the RC chapter in this book so if you need information on creating RC files and using them you should refer to Chapter 3, “Using the Resource Files to Change the Desktop” on page 19.

After getting the CID server installed and testing the normal CID install to insure everything was working we used the UserExit, in the response file, to call a CMD file we had created. We tested a couple of different versions of the CMD file which will be discussed in this section. The first version of the CMD file we tested is displayed in Figure 167.

```
COPY X:\SMALL.RC C:\OS2
C:
CD\OS2
ERASE OS2.INI
MAKEINI OS2.INI SMALL.RC
```

Figure 167. Create New INI

This version of the cmd file will:

1. Copy a customized RC file, from the CID directory on the server (X:), to the OS2 subdirectory on the boot drive of the target system
2. Go to the C drive (install drive) on the target system
3. Erase the existing OS2.INI file from the target system
4. Use MAKEINI to create the new OS2.INI file from the customized RC file

The second CMD file we tested is displayed in Figure 168 and it is basically the same as the first except we removed the line that erased the existing INI file. This resulted in the desktop coming up with the standard objects from install as well as the objects that were in the SMALL.RC file.

```
Copy X:\SMALL.RC C:\OS2
C:
CD\OS2
MAKEINI OS2.INI SMALL.RC
```

Figure 168. Update INI

The UserExit is the last step in a CID installation process to be executed, during the installation, prior to the target system being booted to complete

the installation. Because of this the INI files will already exist on the hard drive so if you want a totally new desktop you must erase the existing INI file prior to executing the MAKEINI command. If you don't erase the existing INI file it will be used as a template and the new INI file will contain the same objects as the old one as well as any new ones defined in the RC file used in the MAKEINI.

We found that if you are trying to simply add to the default desktop, by not erasing the existing INI file prior to the MAKEINI, any object in the customized RC file that already exists somewhere in the default INI file will not get created on the new desktop. An example of this is our Small.RC file containing a desktop which had only six objects in it. These objects were:

1. OS/2 System Editor
2. OS/2 Windowed Command Session
3. Calculator
4. Pulse
5. Cat and Mouse
6. Solitare - Klondike

When we used the Small.RC file during the MAKEINI, after erasing the existing OS2.INI file, our newly installed system came up with only those 6 objects on the desktop. If we did not erase the OS2.INI file prior to running the MAKEINI, our new desktop would contain only the default desktop objects. The six objects from the Small.RC file did not show up on the desktop.

We then edited the Small.RC file and changed the OBJECTID of each of the 6 objects, we simply added the number 2 to each of the existing IDs. We then repeated the installation, using the CMD file that did not erase the OS2.INI, and the desktop that came up contained the default objects, as well as the additional six objects from the Small.RC file.

5.3 Customization of INI

We tested two ways of customizing your INI files during the CID install:

- Using REXX
- OS2IniData keyword

The REXX method allows you to access most of the data in the INI file and because of this it is a more versatile method.

5.3.1 REXX the INI Files

We tested using REXX to customize your INI files during a CID install of OS/2. This procedure requires the system to be up and running prior to the REXX program executing.

Note

You must have REXX installed on your target system or you cannot use a REXX program to update the INI files.

We found two ways of getting REXX to update your INI files:

1. Using UserExit and CMD files containing the REXX procedures needed
2. Using the CID client CMD REXX file to call the REXX procedure

5.3.1.1 REXX via the UserExit

The first step, in getting REXX access to the INI files during a CID install, is to set up a UserExit from within the response file you are using for the install. The UserExit we used was **UserExit=X:\03.CMD** where X: is the the redirected drive on the CID Server and 03.CMD was the file with the procedures we wanted to run. The contents of the 03.CMD file are displayed in Figure 169.

```
Copy X:\STARTUP.CMD
Copy X:\INICHG.CMD
exit
```

Figure 169. 03.CMD File

This command file simply copied two files from the CID server to the target system's boot drive. The contents of these two files are shown in Figure 170 on page 206 and in Figure 171 on page 206. The explanations of these files will follow each of the figures.

```
/* REXX program to set contextmenu mouse mapping to */
/* Shift and mouse button 1 */
call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'
call Sysini 'USER','PM_ControlPanel','ContextMenuMouse','13040800'x
exit
```

Figure 170. INICHG.CMD

The INICHG.CMD program is a simple REXX program which remaps the settings in the INI file so that, in order to get the System menu for Shutdown, Lockup, etc., you will now have to hold the Shift button down and then press mouse button 1 while the pointer is on the PM background.

```
/* REXX program that causes the STARTUP.CMD to Sleep */
/* for 45 seconds to allow the system enough time to boot */
call RxfuncAdd 'SysSleep','RexxUtil','SysSleep'
call SysSleep 45
'start INICHG'
'erase INICHG.CMD'
'erase STARTUP.CMD'
'SETBOOT /IBA:CID'
```

Figure 171. STARTUP.CMD

The STARTUP.CMD file is the one that actually calls the INICHG.CMD as well as cleans up after itself and reboots the system. Since the system is being started for the first time, after a new install, it takes a little while longer to come up then normal. This would cause the STARTUP.CMD file to execute before the system was really ready for any inputs to the INI file. To get around this problem we used the REXX SysSleep function to pause the execution of Startup for 45 seconds. Then the INICHG file is called which makes the update to the INI. At this time the cleanup is done where the INICHG.CMD and the STARTUP.CMD files are removed from the hard drive. In order for the changes to take effect the system has to be rebooted so we used the SETBOOT command to reboot the system to the partition that we had just finished installing.

This procedure should work for making any updates to the INI file that REXX is allowed to do. We then tested making several changes to the INI file in the same procedure. The INICHG file for this test is displayed in Figure 172 on page 207 and the comments included in the file explain what things are being changed.

```
/* REXX program to make updates to the INI file */

call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* Register SysIni */

call Sysini 'USER','PM_ControlPanel',,   /* Set context mouse mapping */
'ContextMenuMouse','13040800'x        /* to Shift and mouse button 1 */

result = SysIni('USER','PM_ControlPanel',,   /* Turn off Animation */
            'Animation','00000000'x)

/* initialize variables */

keyname = 'Beep'
appname = 'PM_ControlPanel'
inifile = 'USER'
value = "0'
NULL = '00'x
value = value || NULL
result = SysIni(inifile, appname,,
            keyname, value)  /* Disable Warning Beeps */

/* initialize variables */

keyname   = 'BorderWidth'
value     = '10'
result = SysIni(inifile, appname,,
            keyname, value)        /* Set default border width to 10 */

exit
```

Figure 172. More Complex INICHG.CMD

This CMD file executed fine and after the reboot, the changes were implemented on our system.

For more information on what types of things you can do with REXX and the INI files you should reference Chapter 4, “Using .INI files to Change the Desktop” on page 119.

5.4 REXX from the LCU Command File

The LAN CID Utility (LCU) procedure allows you to call additional REXX procedures and execute these as part of the CID install. This is true even if the additional procedure requires the system to do a reboot before and/or after. The way to do this is to add some additional definitions and DoForever statements to the client LCU CMD file the CID procedure executes during a normal install. Examples of the Product definitions that we needed are shown in Figure 173 and in Figure 174 on page 209.

```
x.thinifs1 =  8
x.8.name =  "SRVIFS Requester1"
x.8.statevar = "CAS_" || x.8.name
x.8.instprog =  img_dir"\srvifs\thinifs /S:"img_dir"\srvifs,
                    " /t:"ifs_dir" /tu:"bootdrive"\",
                    "/l1:"log_dir"\srvifs\"client".log",
                    "/req:"client" /srv:"srvifs_alias1" /d:"d1
x.8.rspdir = ""
x.8.default = ""

x.thinifs2 =  9
x.9.name =  "SRVIFS Requester2"
x.9.statevar = "CAS_" || x.9.name
x.9.instprog = img_dir"\srvifs\thinifs/S:"img_dir"\srvifs/t:"ifs_dir",
                    "/tu:"bootdrive"\",
                    "/l1:"log_dir"\srvifs\"client".log",
                    "/req:"client" /srv:\\CODESRV1\LCULOG /d:"d3
x.9.rspdir = ""
x.9.default = ""
```

Figure 173. Sample Product Definitions

```
x.casdelet =  11
x.11.name =  "LAN CID Utility Delete"
x.11.statevar = "CAS_" || x.11.name
x.11.instprog =  img_dir"\lcu\casdelet /pl:"dll_dirs" /tu:"bootdrive
x.11.rspdir = ""
x.11.default = ""

x.casinstl = 12
x.12.name='LCU'
x.12.statevar = "CAS_" || x.12.name
x.12.instprog='x:\img\lcu\casinstl',
                  '/cmd:rsp_dir"\client',
                  ' /tu:' ||bootdrive || ' ',
                  ' /pl:rsp_dir\dll;rsp_dir:\img\lcu; ',
                  ' /pa:rsp_dir\img\lcu',
                  ' /l1:log_dir\lcu\' || client || '.log ',
                  ' /l2:log_dir\lcu\SRVIFS_REQ.log', ' /D'
x.12.rspdir   = ''
x.12.default  = ''

     /* Procedure to change the INI file */

x.inichg = 13
x.13.name='INI CHANGE'
x.13.statevar = ''
x.13.instprog = 'x:\INICHG.CMD'
x.13.rspdir   = ''
x.13.default  = ''

x.thinifs3 =  14
x.14.name =  "SRVIFS Requester3"
x.14.statevar = ""
x.14.instprog =  img_dir"\srvifs\thinifs /S:"img_dir"\srvifs,
                    " /t:"ifs_dir" /tu:"bootdrive"\",
                    "/l1:"log_dir"\srvifs\"client".log",
                    "/req:"client" /srv:\\CODESRV1\LCULOG /d:"d2
x.14.rspdir = ""
x.14.default = ""

NUM_INSTALL_PROGS = 14
```

Figure 174. Second Part of Product Definitons

The Thinifs definitions will place the following LCU redirector files on the target hard disk:

- IFSDEL.EXE
- SRVIFS.SYS
- SRVIFSC.IFS
- SRVATTCH.EXE
- XLI.MSG
- XLIH.MSG

It also updates the path and adds several CALL statements to the CONFIG.SYS, so that the target system will still be able to connect to the CID server after it reboots.

The Casinstl will install LCU on the hard disk of the target system and it will also create a STARTUP.CMD that will execute after the reboot. The Casdelet will remove the LCU, the LCU redirector files, the STARTUP.CMD and the CONFIG.SYS statements that where added earlier.

The INICHG.CMD is the REXX procedure that we are using to modify the user INI file on our target system (see Figure 175 on page 211).

```
.
* REXX program to make updates to the INI file */

call RxFuncAdd 'SysIni', 'RexxUtil', 'SysIni'  /* Register SysIni */

call Sysini 'USER','PM_ControlPanel',,    /* Set context mouse mapping */
'ContextMenuMouse','13040800'x         /* to Shift and mouse button 1 */

result = SysIni('USER','PM_ControlPanel',,    /* Turn off Animation */
            'Animation','00000000'x)

keyname = 'Beep'
appname = 'PM_ControlPanel'
inifile = 'USER'
value = '0'
NULL = '00'x
value = value || NULL
result = SysIni(inifile, appname,,
            keyname, value)  /* Disable Warning Beeps */

keyname   = 'BorderWidth'
value     = '10'
result = SysIni(inifile, appname,,
            keyname, value)        /* Set default border width to 10 */

keyname = 'Menus'
appname = 'PM_SystemFonts'
value = '14.Times New Roman'

result = SysIni(inifile, appname,,
            keyname,value)       /* Set default font for all system menus
 */
                                          /* to 14 point Times New Roman *
/
exit
```

Figure 175. Example of Our INICHG.CMD

The comments within the procedure describe what each section is intended to modify. This is the same CMD file that we used in the earlier sections of this book and it was a proven good procedure so it is the first one we tested this CID REXX CMD file procedure with.

We have included our DoForever loop in Figure 176 on page 212. The DoForever is the actual order that the various procedures are going to be

executed. The Call CheckBoot will allow the system to go through the basic OS/2 install, the laps install, and the three thinifs installs and then reboot as many times as required in order to get the remaining procedures executed. These reboots are needed so that the Thinifs gets set up, the INI CHANGE segment executes, the Casdelet is executed and cleans up the target system, then the system is actually started with the modified INI files.

Warning

If you have defined Reboot Required=1 in the response file the Checkboot will not be executed and the CID procedure will terminate after the base OS/2 code has been installed and the system reboots. You must have Reboot Required=0 in your response file in order for the REXX update to be done.

```
Do Forever
  Select
    When OVERALL_STATE = 0 then do
      if RunInstall(x.seinst)  == BAD_RC then exit
      if RunInstall(x.laps)  == BAD_RC then exit
      if RunInstall(x.thinifs1)  == BAD_RC then exit
      if RunInstall(x.thinifs2)  == BAD_RC then exit
      if RunInstall(x.thinifs3)  == BAD_RC then exit
      if RunInstall(x.casinstl)  == BAD_RC then exit
      Call CheckBoot
    end
    When OVERALL_STATE = 1 then do
      if RunInstall(x.inichg)  == BAD_RC then exit
      if RunInstall(x.ifsdel)  == BAD_RC then exit
      if RunInstall(x.casdelet)  == BAD_RC then exit
      Call Reboot
    end
  end
end
```

Figure 176. Sample of DoForever Loop

5.4.1 OS2IniData Keyword

There is another way to make some updates to your system during a CID install. This is using the OS2IniData keyword to write new values into your OS2.ini file during the install. However, this is limited due to the fact this keyword makes use of the PrfWriteProfileString API which has some limitations:

- It will not write HEX data and a lot of the INI values are HEX data.
- It will not create an object.
- It will not update a value if the value is not already listed.

An example of the third limitation above are the system fonts. If you do not make any changes to the system fonts there will not be a listing for them in the INI file. This is because the system fonts are hard coded into the system and there is no need for them to be in the INI file until you decide to make changes to them. At that time the INI file will be updated, with the information on the font you add or change, and the INI will override the hard coded info at boot time. After this is done the PrfWriteProfileString API would be able to write changes to the INI file. However at the time of our CID installation the system will still be running with the defaults so there will be no listing for Fonts in the INI file and the API will not work. This means the only things you can update using this method are ones that don't use HEX values as updates and are actually listed in the INI file at install time. This turned out to be a rather limited list as shown in Table 50 on page 214. The following lines were tried out with varying results:

```
OS2IniData=/PM_ControlPanel/Animation/'00000000'x/

OS2IniData=/PM_ControlPanel/ContextMenuMouse/13040800/
```

We found that the Animation line, which actually had the HEX value in it, was ignored and had no effect on the system. The ContextMenuMouse line value had decimal numbers in it and even though it was not the correct value for this item, the INI was actually modified. However we have no idea what the value was indicating to the ContextMenueMouse, but it did disable the normal mouse button #2 access to the menu. This could be useful if you wanted to totally eliminate mouse access to the ContextMenue.

One example of a value that can be successfully modified with OS2IniData is the BorderWidth. We used the following line in our Response file to set the borders on our system to 10 rather then the default 4:

```
OS2IniData=/PM_ControlPanel/BorderWidth/10/
```

Table 50. Examples of Options for OS2IniData

AppName	KeyName	Values
PM_Colors	See Table 51 on page 215	Three values ranging from 0 - 255
PM_ControlPanel	BorderWidth	1 to what ever you like
PM_Default_Colors	See Table 50 on page 214	Three values ranging from 0 - 255
PM_National	iCountry	See Table 35 on page 149
PM_National	iCurrency	See Table 36 on page 150
PM_National	iDate	See Table 37 on page 152
PM_National	iDigits	See 0 to 3
PM_National	iLzero	1 for yes or zero for no
PM_National	iMeasurement	See Table 38 on page 156
PM_National	iTime	See Table 39 on page 157
PM_National	s1159	am or AM or 2 spaces
PM_National	s2359	pm or PM or 2 spaces
PM_National	sCurrency	Maximum 3 characters identifying currency
PM_National	sDate	1 character which separates day, month and year
PM_National	sDecimal	1 character which separates whole and fraction parts of numbers
PM_National	sList	1 character which separates items in a series
PM_National	sThousand	1 character which separates every 3 digits in a number
PM_National	sTime	1 character which separates hours, minutes and seconds

Table 51. PM_Colors KeyNames

ActiveBorder	HelpText	OutputText
ActiveTitle	HiliteBackground	PageBackground
ActiveTitleText	HiliteForeground	Scrollbar
ActiveTitleTextBgnd	IconText	Shadow
AppWorkspace	InactiveBorder	ShadowHiliteBgnd
ButtonDark	InactiveTitle	ShadowHiliteFgnd
ButtonDefault	InactiveTitleText	ShadowText
ButtonLight	InactiveTitleTextBgnd	TitleBottom
ButtonMiddle	Menu	TitleText
DialogBackground	MenuDisabledText	Window
EntryField	MenuHilite	WindowFrame
FieldBackground	MenuHiliteText	WindowStaticText
HelpBackground	MenuText	WindowText
HelpHilite		

One thing of note on using this keyword is that anything that you can update with this keyword can also be updated via the RC files. So if you are going to be making a large number of changes to these type of things, and they are all going to be the same on every workstation, you may want to look at using the RC method which was discussed earlier. However, if you only want to make a few changes, or if you are making different changes to each workstation, it is a relatively simple procedure to enter a OS2IniData line for each of these in the appropriate Response file prior to starting the installation. In this way you have the flexibility to quickly change the customization you are going to apply each of your target systems.

5.5 Combined Scenario

We believe the majority of people will probably use the RC file method of updating their desktops during a CID install; however there are things that some administrators may wish to modify that can not be done via the RC files. These would be any thing that has HEX values in the INI file. One example of this would be setting the ContextMenuMouse so that a user could not longer access the ContextMenu by pressing mouse button #2 on the desktop background. This could be useful in making the desktop less accessible to users who are not familiar with its operation, and some of the perils of making changes to it.

What we decided to do was use a combination of the RC and the REXX CMD file methods of modifying the INI file on our target system. We once again used our INICHG.CMD REXX file and the Small.RC file. The RC file was used in the same manner as explained in the 5.2, “RC Files” on page 203 section.

The INICHG file was implemented with the same procedure as was described in 5.4, “REXX from the LCU Command File” on page 208.

Running the two procedures during the same CID installation worked fine and we had no errors reported. When the target system was brought up it had the 6 objects from the small RC desktop and the changes that the Inichg file implemented where there as well. The borders were bigger, the animation was turned off, etc.

The one final procedure you may want to bring into this scenario would be the OS2IniData keyword from the response file. This also works fine along with the RC and the INI file procedures.

5.6 CID Conclusions

The conclusions we arrived at after the CID tests we have done are:

- There are an extensive number of changes you can make to a target system during a CID install.
- It provides you with several ways of controlling how these changes are done and what the desktop will look like and what you will be able to do from it.
- There are several choices of which procedure to use when implementing your changes.
- There are several combinations of the procedures that work.
- It allows an administrator to implement one customized desktop for all users in his network.
- It provides the needed flexibility so that some systems can be installed with some individual differences from the rest.

We believe that the majority of administrators will decide on one method of implementing the customization during a CID installation that they are most comfortable with. However, we have provided the possibility of using various combinations of procedures in order to get the exact results you are looking for, in the easiest manner possible.

Appendix A. ITSO DESKTOPS Utilities

Warning!

Before using the supplied programs to create and switch between desktops, read this entire section! Pay especially close attention to the problems described in A.2, “Deleting Desktops” on page 222.

These programs were written as tools to help us test the .RC and .INI file changes we have described in the preceding chapters. We thought they might be useful to you for the same reasons. We have provided the source for these programs as well.

They are meant to be tools and examples and not to be used in critical environments. Because we know you may attempt to use them as such, please test them thoroughly before introducing them into a production environment.

We mainly hope that they will save you time in creating desktops and by not having to encounter the same problems and mistakes that we did.

The objects and settings for a desktop are always stored in the user.INI file. The associated directory structure is stored on the boot drive. After installing OS/2 the default user.INI file is OS2.INI and the directory structure is DESKTOP on the boot drive.

To switch to another desktop you only need to point to a new user.INI file. If the user.INI file you define has never been used before on the system then a new directory structure for the desktop will be created on your boot drive. If the user.INI file was used before with OS/2 on *any* system, then the associated directory structure has to be present on the boot drive in order for it to work properly.

The connection of the user.INI file and the directory structure is done by using an application and key in the user.INI file. The user.INI file has an application PM_Workplace:Location and a key <WP_DESKTOP> that contains the file system object handle of the associated desktop directory on the boot drive.

Switching desktops can be done in two different ways:

- From the CONFIG.SYS using SET USER_INI=

- From a program using PrfReset

Using the SET USER_INI= statement in the CONFIG.SYS to point to another user .INI file requires a reboot of the system to take effect. These .INI files can be created by following the methods described in Chapter 3, “Using the Resource Files to Change the Desktop” on page 19.

For switching desktops ″on the fly″ there is a PM API call, **PrfReset**. With this call you can point to a new user.INI file. After a call to this API, all desktop objects and the current desktop itself will be closed and a new desktop will be active and displayed. Because of this you can:

- Create multiple desktops on the same machine
- Move desktops from one machine to another

Warning!

With PrfReset you can only point to a user.INI file. Using a system.INI file will hang your system.

A.1 Using PrfReset

For switching desktops we wrote the program **CHGDT** that uses the PM API call PrfReset to change to another user.INI file. When calling the program without any parameter, it displays the actual system and user INI file and the desktop directory name. To query the name of the directory we had to write a dummy SOM object because some SOM calls can only be used within a SOM object. Before you run CHGDT, you have to install the **QDESK.DLL** with the command **INSTALL**. This will copy the QDESK.DLL to the OS2DLL directory on the boot drive and register the class GEQueryDesktop. When calling CHGDT with the new of a new user.INI file, the program will check if it is a valid user.INI file. Then it switches to the new user.INI file by calling PrfReset. The program CHGDT also has a parameter, **/C** that will update the CONFIG.SYS so that the new desktop will be used after the next reboot.

If the user.INI file was used by OS/2 before, then the program will also check if the associated directory structure is available on the system. If this is the first time this user.INI file has been used then a new directory structure will be created by the system.

Because we ran into a few problems with the PM API call PrfReset, here is a description how we solved the problems. The PrfReset call uses a data

structure with two data pointers. There are two ways to fill the data structure (see Figure 177 on page 220).

We wrote two programs to show you how to, and how not to, fill the data structure. Version 1 of the data structure is used in the program in Figure 179 on page 221. We found that this causes problems and hangs the system on the second call of the program. Version 2 of the data structure, used in the program in Figure 180 on page 222, worked much better. We could call this program many times without any problems. We think that OS/2 or the API has problems if the complete structure (structure and fields) are not in the same contiguous memory area.

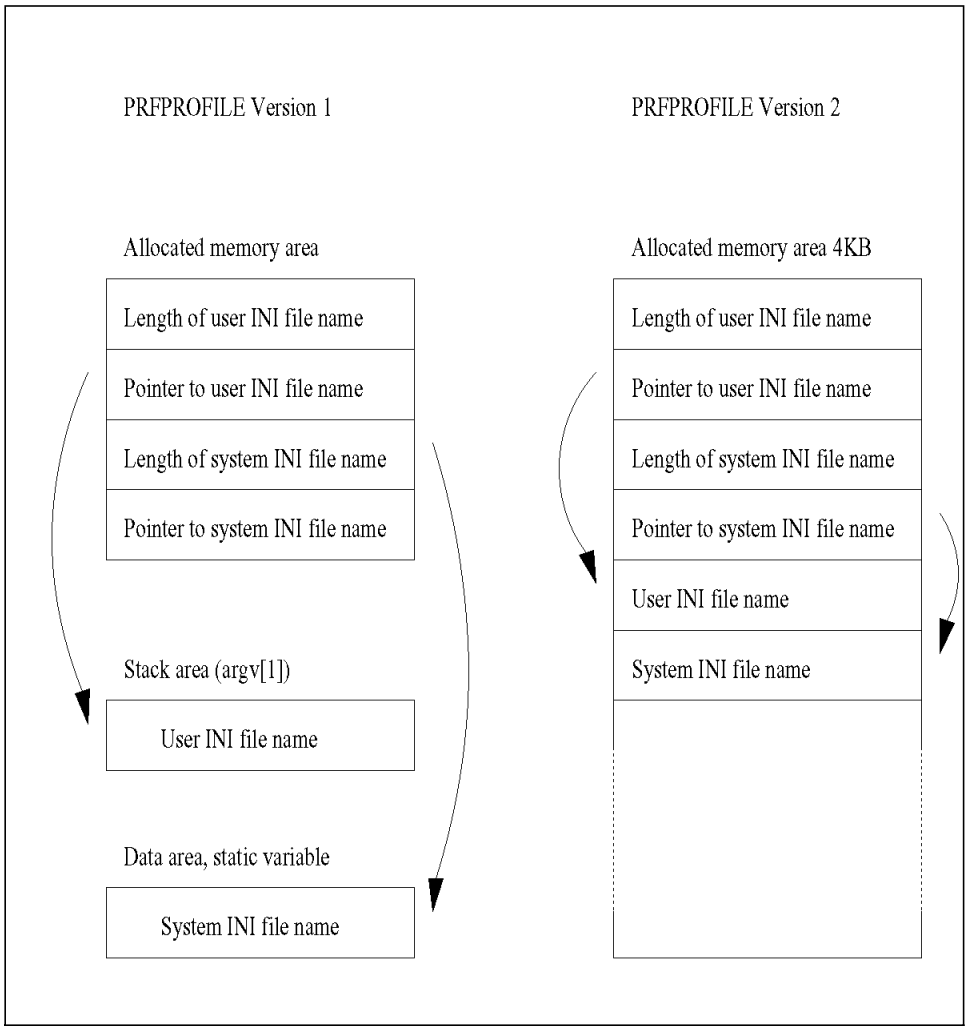


Figure 177. PrfReset Data Structure

```
typedef struct _PRFPROFILE    /* prfpro */
{
   ULONG  cchUserName;
   PSZ    pszUserName;
   ULONG  cchSysName;
   PSZ    pszSysName;
} PRFPROFILE;
typedef PRFPROFILE *PPRFPROFILE;
```

Figure 178. PrfReset API structure from OS/2 Toolkit (PMSHL.H)

```
        PRFPROFILE prfProfile;
        HAB        hab;
        BOOL       fSuccess;

prfProfile.pszUserName = argv[1];
prfProfile.cchUserName = strlen(prfProfile.pszUserName);
prfProfile.pszSysName = "C:\\OS2\\OS2SYS.INI";
prfProfile.cchSysName = strlen(prfProfile.pszSysName);

fSuccess = PrfReset(hab, &prfProfile);
```

Figure 179. PrfReset with Problems

```
        PPRFPROFILE pprfProfile;
        HAB         hab;
        BOOL        fSuccess;
        APIRET      rc;
        PSZ         Ptr;

rc = DosAllocMem(&Ptr, 4096,
                 PAG_COMMIT | PAG_READ | PAG_WRITE);

pprfProfile = (PPRFPROFILE)Ptr;

pprfProfile->pszUserName = (PSZ)&pprfProfile[1];
strcpy(pprfProfile->pszUserName, argv[1]);
pprfProfile->cchUserName = strlen(pprfProfile->pszUserName) + 1;
pprfProfile->pszSysName =
    &pprfProfile->pszUserName[strlen(pprfProfile->pszUserName) + 1];
strcpy(pprfProfile->pszSysName, "C:\\OS2\\OS2SYS.INI");
pprfProfile->cchSysName = strlen(pprfProfile->pszSysName);

fSuccess = PrfReset(hab, pprfProfile);
```

Figure 180. PrfReset that Works More Than Once

A.2 Deleting Desktops

When you switch between desktops only the user.INI file is changed. The system.INI remains the same. This means that entries are made in the system.INI file for each and every user.INI file that has been active.

It is here that we encountered a problem when trying to delete desktop directory structures that were associated with an active system.INI file. In most cases the pointer to the desktop in the active user.INI file was deleted by the system. This meant that the next time we booted the system it could not find the active desktop.

It is for this reason that we say that if you want use this set of programs in a production environment please be sure to test it thoroughly. Once you start creating desktops the only way we found to effectively remove them is to boot the system from diskette or another partition and then proceed with the deletion.

There are several utilities for deleting entire directory structures, such as desktop directory structures, and we have provided you with our own. It is

called DELALL and will delete all the files and remove all of the directories *including* the specified directory. For example,

```
DELALL C:\DESKTOP
```

will remove the C:\DESKTOP directory and all of the files and directories underneath it. If you want to delete all of the files and directories but do not want to remove the desktop directory itself, then you can use

```
DELALL
C:\DESKTOP\*.*
```

We used the *.* method in our testing as the .INI files remained stable as long as we did not remove the base desktop directory of our new desktop.s.

A.3 Managing Multiple Desktops on One Machine

In order to create and switch between desktops on the same machine, we have written a user interface program using VisPro/REXX. When you start the program it will display all valid user.INI files found in the OS2 directory of the boot drive. In the case of a new installation or a system where there have been no new .INI files built, the selection box will be blank (see Figure 181 on page 225).

If you click on New, a file dialog will come up where you can select a .RC file to create or update a .INI file in the OS2 directory on the boot drive (see Figure 182 on page 225). You can use the supplied name for the .INI file or, in the case of compiling over an existing .INI file, enter an alternative name (see Figure 183 on page 226). This new .INI file will then be added to the list and available for selection (see Figure 184 on page 227).

After selecting one of the .INI files you can then click on Switch to switch to that .INI file and associated desktop. You will be prompted if you want to update the CONFIG.SYS with this new .INI file. If you click on YES, then the next time you boot your machine it will use this new desktop (see Figure 185 on page 227).

The first time you use a .INI file a directory structure will be created for it. You can see this by looking at the difference between Figure 186 on page 228 and Figure 187 on page 228. In one case the directory structure already existed and in the other case it did not.

A.3.1 Installing the DLL

Before you use the Desktops program you must first register a DLL. This is done by running the INSTALL.CMD program. We do this automatically when you open the Desktops program. The command call is in the When Opened section of the Main form of the VP-REXX program. Since you only need to do this once on each machine that will be using the Desktops program you may want to change this for your environment. Trying to reload the DLL every time you start the program, however, will not create a problem.

A.3.2 Creating a New Desktop

To create a new desktop from a .RC file do the following:

1. Double click on the Desktops program icon if the program is not already started
2. Click on **New** (see Figure 181 on page 225)
3. Click on the .RC file you want to create the desktop from (see Figure 182 on page 225)
4. Click on **OK**

   You should get an information dialog saying the .INI file was created successfully (see Figure 183 on page 226).
5. Click on **OK**

The .INI file should then be added to the list box of the Desktops program (see Figure 184 on page 227).

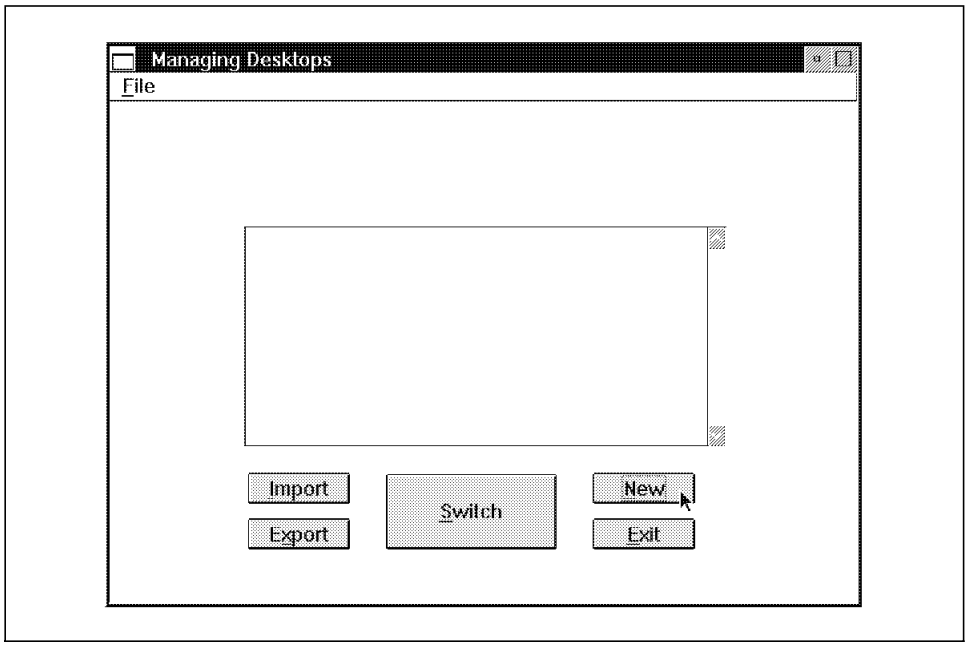


Figure 181. DESKTOPS Main Selection Dialog

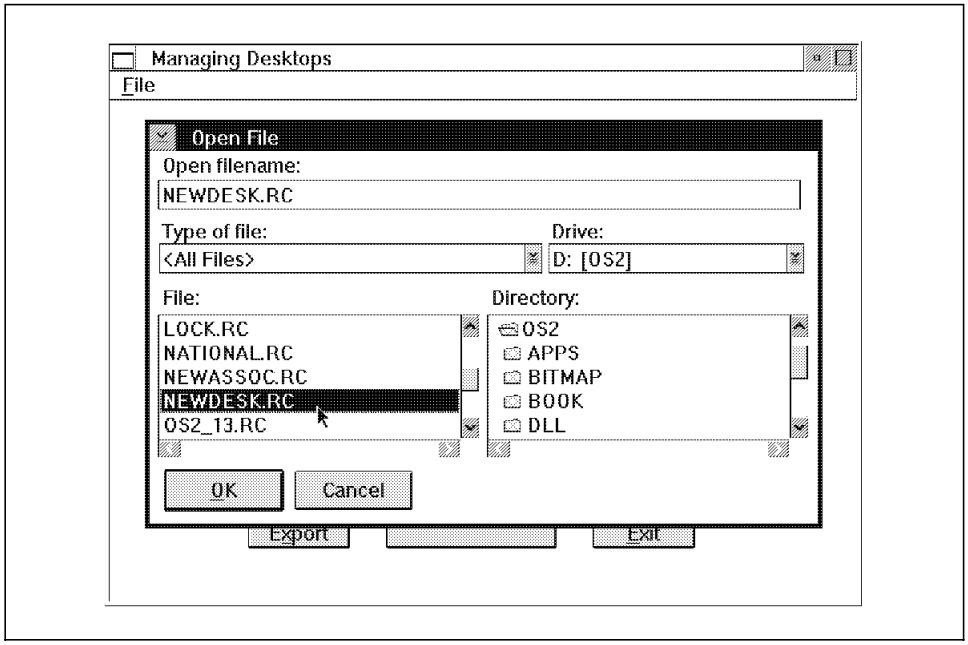


Figure 182. Using New to Make a .INI File From a .RC File

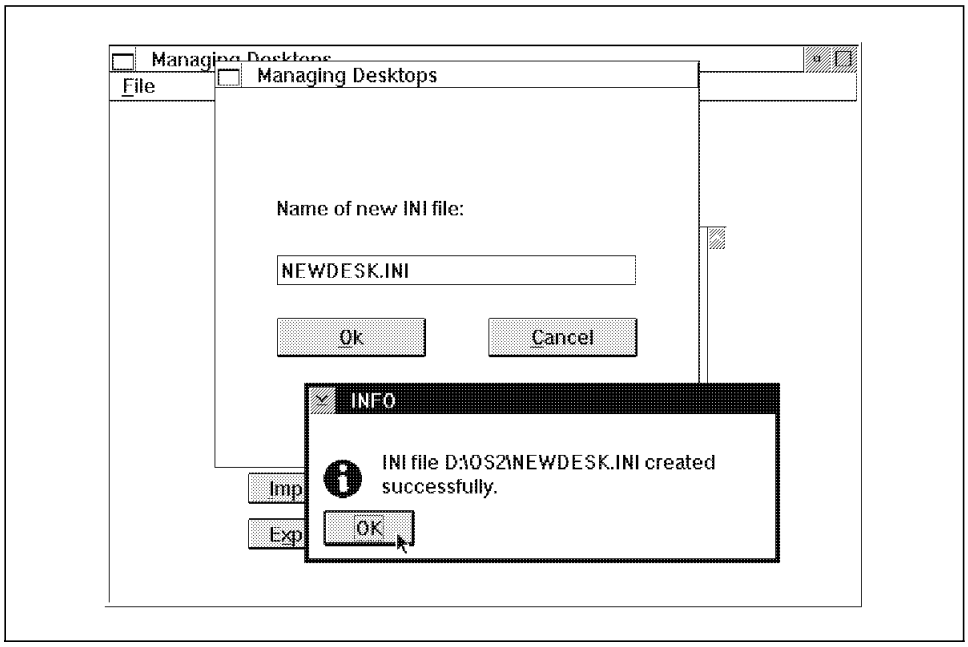


Figure 183. Prompt After Successful Completion

A.3.3 Switching Desktops

To switch desktops do the following:

1. Double click on the Desktops program icon if the program is not already started
2. Click on the .INI file you want to switch to
3. Click on **Switch** (see Figure 184 on page 227)
4. You will be prompted if you want to update the CONFIG.SYS file

   If you click on **Yes** then the desktop you are switching to will be used the next time you boot your system. If you click on **No** then the current .INI file in the CONFIG.SYS file will be used when you reboot your system (see Figure 185 on page 227).
5. Wait for all disk activity to stop
6. Click on **OK** (see Figure 186 on page 228 and Figure 187 on page 228)

The new desktop should now be displayed and the .INI file for that desktop will be removed from the list box of the Desktops program. The .INI file of the desktop you just switched *from* should be added to the list.

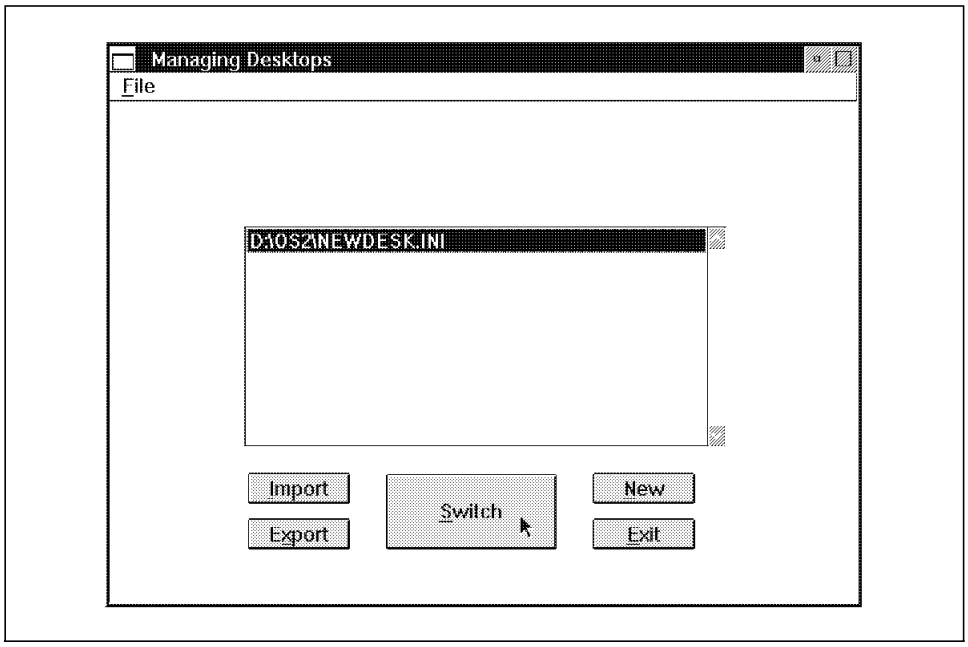


Figure 184. Switching to the New Listed File

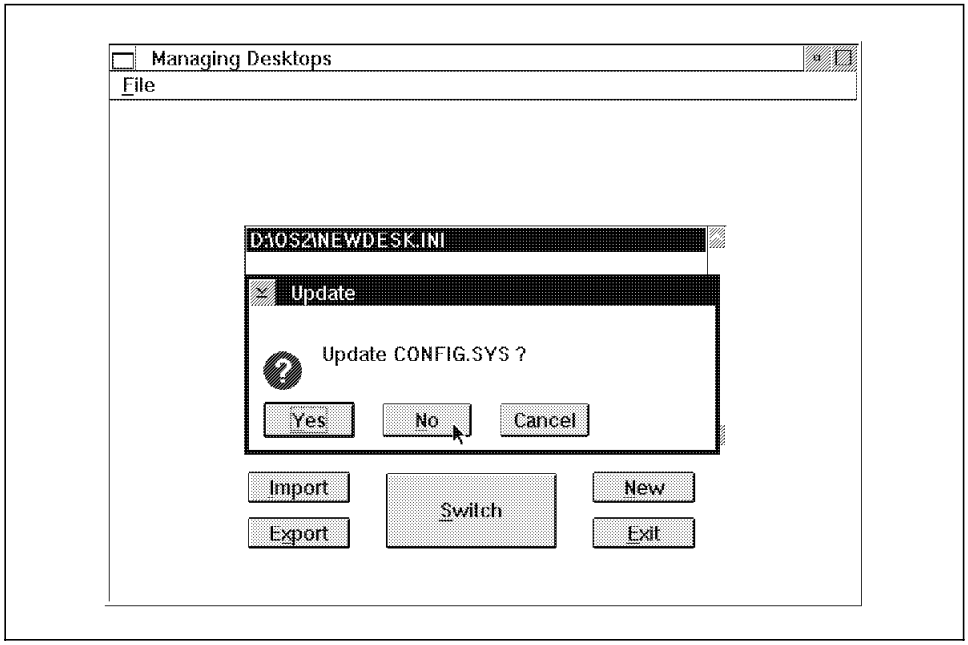


Figure 185. Prompt for Updating the CONFIG.SYS

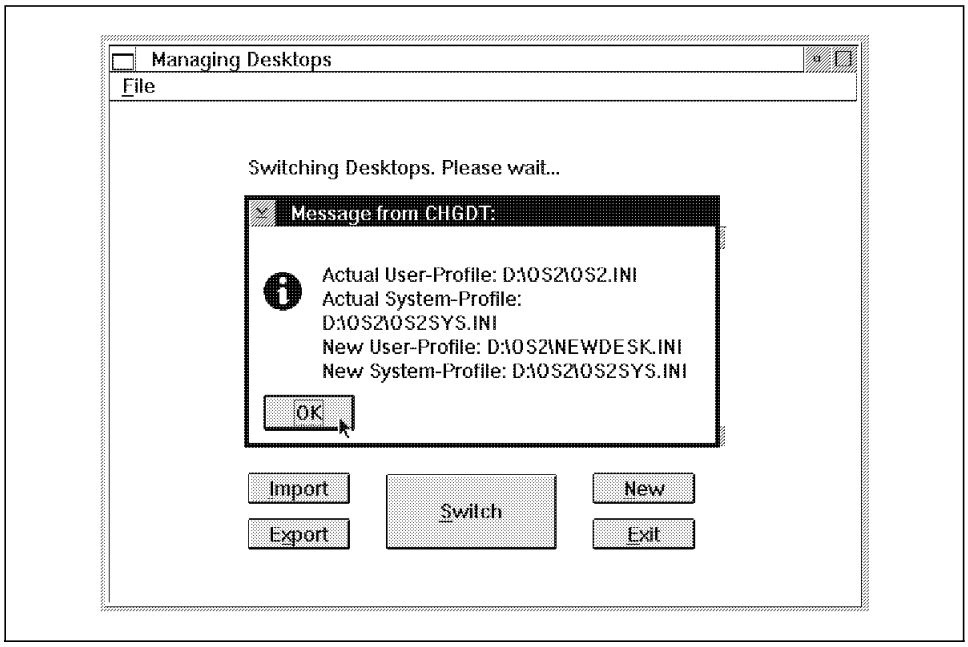


Figure 186. Information Dialog if this is a New Desktop

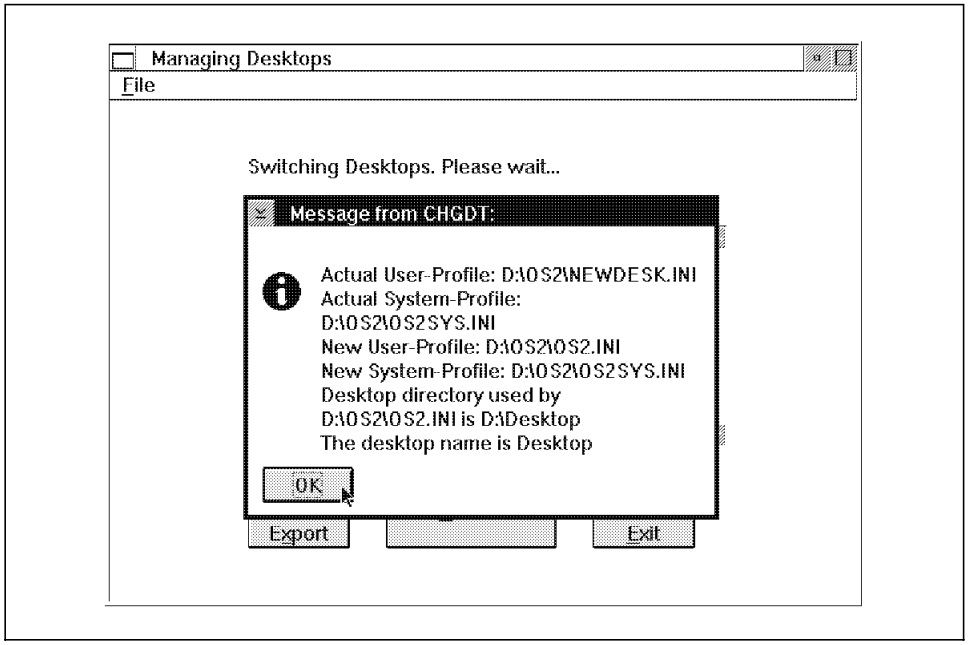


Figure 187. Information Dialog if the Desktop Already Exists

A.4 Moving Desktops from One Machine to Another

You can also move a desktop and its associated desktop from one machine to another by using some of the programs provided on the enclosed diskette. We have hooked these programs into the visual REXX program we described previously and have also provided a command line description if you choose to do this in another way.

A.4.1 Exporting a Desktop Using the Desktops Program

You can export a desktop from a machine by selecting a .INI file from the list box and simply clicking on the Export button of the Managing Desktops program. You will then be prompted to select a Drive and Path for the desktop. This can be any defined drive on your system. That is, local, diskette or even a network drive (see Figure 188).

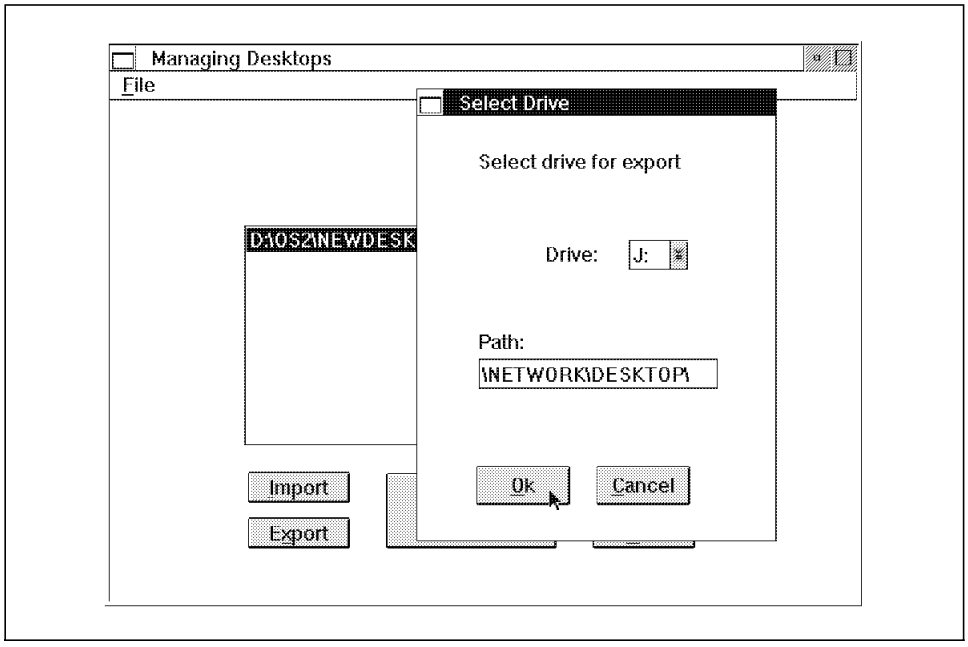


Figure 188. Exporting a Desktop

A.4.2 Importing a Desktop Using the Desktops Program

After you have exported a desktop as described in A.4.1, “Exporting a Desktop Using the Desktops Program” on page 229 you can then import it to another machine by clicking on the Import button of the Managing Desktops program. You will then be prompted to select a Drive, Directory and .INI file for import. This .INI file can be copied from any defined drive on your system. That is, local, diskette or even a network drive (see Figure 189). In addition to just copying the files to the new system, the program will also update the required handles in the OS/2 .INI files.

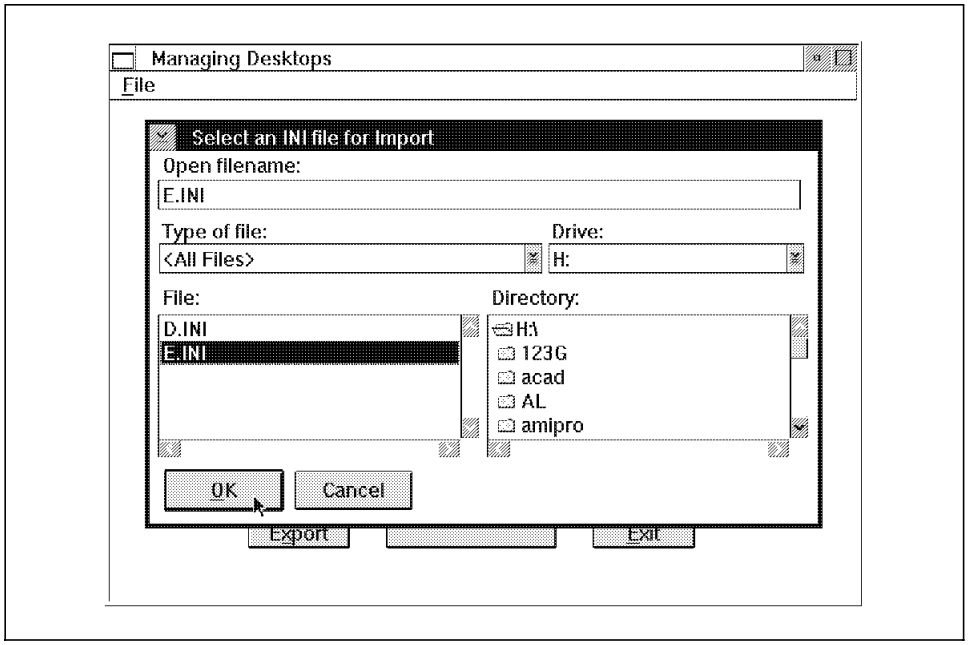


Figure 189. Importing a Desktop

A.4.3 From the Command Line

You can copy a desktop from one machine to another via the command line by copying the .INI file and the desktop directory structure, but there are a few things you need to prepare. All the file system objects have a file system object handle. When copying the desktop structure, for example C:DESKTOP, to another machine, the file system objects will get another file system object handle as on the original machine. Because there are entries with the file system object handles in the user.INI file, all these entries have to be updated on the new machine.

The following applications and keys that are in the user.INI file have to be updated before using the user.INI in another machine:

Application PM_Workplace:Location
The keys within this application contain an object handle to any system object. All handles, that are handles of a file system object, have to be updated on the new machine.

Application PM_Abstract:FldrContent
The keys within this application are object handles, the contents of these entries are the handles of the abstract objects within this folder. To update these entries, the keys have to be renamed to the new object handles.

For the update of these keys in the user.INI, information about the file system object handles on the original machine is needed. For that we were writing the program **DTUPD** that can be used to query the objects on the original machine and update the copied user.INI on the target machine. The query option creates a control file that is used to do the update on the target machine.

```
To query the objects:

    DTUPD /Q [/I:ini file] [/C:control file]

      ini file       Name of the user.INI file that will be
                     used for the query, default: actual user.INI
      control file   Name of the control file that will be used
                     for the update of the new machine,
                     default: STANDARD OUTPUT

To update the user.INI on the target machine:

    DTUPD /U /C:control file /I:ini file [/D:drive]

      control file   Name of the control file created on the
                     original machine.
      ini file       Name of the user.INI file that should be
                     updated.
      drive          If the desktop directory structure is on another
                     disk as on the original machine, the drive letter
                     can be overwritten with this parameter.
```

Figure 190. Syntax of DTUPD

To copy a desktop from one machine to another, do the following:

1. If the desktop that you want to copy is active, switch to another desktop by using the program CHGDT.
2. Copy the user.INI file onto a diskette.
3. Copy the desktop directory structure to the diskette with the *XCOPY* command using the parameters /S /E, for example
   XCOPY C:DESKTOP2 A:DESKTOP2 /S /E
4. Run the program *DTUPD* with the /Q parameter to create a control file, for example
   DTUPD /Q /I:test.INI /C:A:test.CTL
5. Copy the user.INI file to the new machine.
6. Copy the directory structure to the new machine with XCOPY and the parameters /S /E
7. Run the DTUPD program to update the copied user.INI file, for example:
   DTUPD /U /C:a:test.CTL /I:C:OS2test.INI
8. Switch to the copied desktop by using the program CHGDT.

A.5 Recovering From a Corrupted Desktop

If you switch from Desktop A to Desktop B and then back to Desktop A, or to another desktop, before the desktop handle for Desktop B is written to the .INI file, then Desktop B can become detached from its .INI file. This will result in a corrupted desktop if you try to boot your system using Desktop B.

You will know if this has happened because the user.INI file for Desktop B will not reappear in the list of available desktops once you switch from it. If this happens then you can still ″save″ the desktop by running the CHGDT.EXE program, with the /D parameter, from an OS/2 command line. You must know the name of the .INI file *and* the name of its desktop. The syntax for CHGDT.EXE is as follows:

```
CHGDT drive:\path\user.INI /D:drive:\desktop
```

For example, if you installed OS/2 2.1 on C: amd were trying to restore the original desktop you would type:

```
CHGDT C:\OS2\OS2.INI /D:C:\DESKTOP
```

This should rewrite the key for the desktop in the .INI file and switch to the desktop. You can then switch back to the previous desktop you were using, if desired.

Warning!

If you have corrupted the desktop that is associated with the user.INI file that is listed in your CONFIG.SYS then you *must* restore it with the previous procedure before shutting down your system. If you do not, when you reboot your desktop will be corrupted.

Appendix B. Associating Your Data Files

Associations are probably the most useful documented undocumented feature of the OS/2 Workplace Shell. With them you can totally revolutionize the way you work. It is possible with associations to decide what you want to work on and let the Workplace Shell figure out which program to load to let you do it.

B.1 What is An Association?

An association is a way to tell the Workplace Shell that a certain program or application belongs to a particular data file. Once you have set up an association between a data file and a program you can simply double click on the data file and the program will be invoked with that data file loaded. The default action for any data file in a standard system is to try to open the OS/2 System Editor with that file.

Multiple programs can be associated with a single data file (or group of data files) and the default program will be executed by double clicking. The other programs will be included on the cascade menu that appears when you click on the small arrow to the right of the open option on the data file's pop-up menu.

There are two ways of setting up associations. One is to associate a particular data file with a particular program. The other is to associate a particular class of data files with a program. Each of these has advantages and disadvantages as we will see.

B.2 Associating a Single File

To associate a particular file with a program you can perform the following steps:

1. Open the settings view for the data file by:
   a. Clicking with mouse button 2 on the data file
   b. Clicking with button 1 on the small arrow next to the Open menu choice (See Figure 191 on page 236)
   c. Selecting the Settings menu choice
2. Click on the menu tab to bring up the menu page

3. Click on the Open entry in the upper listbox to show the possible entries on the open cascade menu (See Figure 192 on page 237)

   It is here that we need to add a new entry for the program we wish to associate with this particular data file (in this case EPM.EXE, the OS/2 enhanced editor).

4. Click on the lower of the two Create Another buttons and fill in the menu name and path of the program to associate.

   For the enhanced editor we will use the menu name "enhanced editor" and the path C:OS2APPSEPM.EXE (See Figure 193 on page 237). If you have installed OS/2 on a different drive or moved the enhanced editor then adjust your path accordingly.

   If you later need to modify this information you can use the lower settings button after you have selected the enhanced editor in the lower listbox.

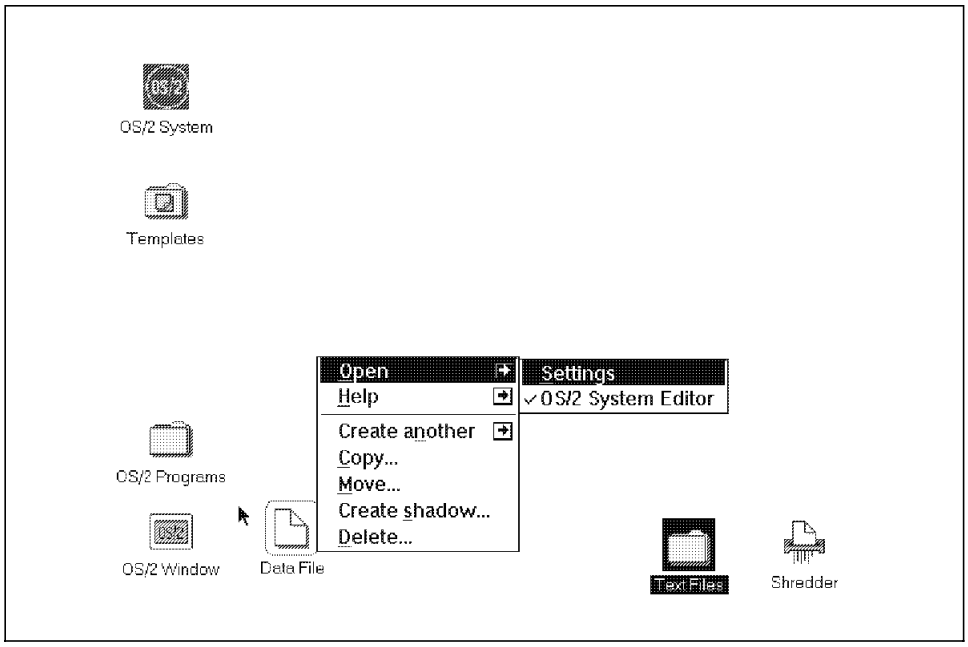


Figure 191. Opening Settings for a Data File

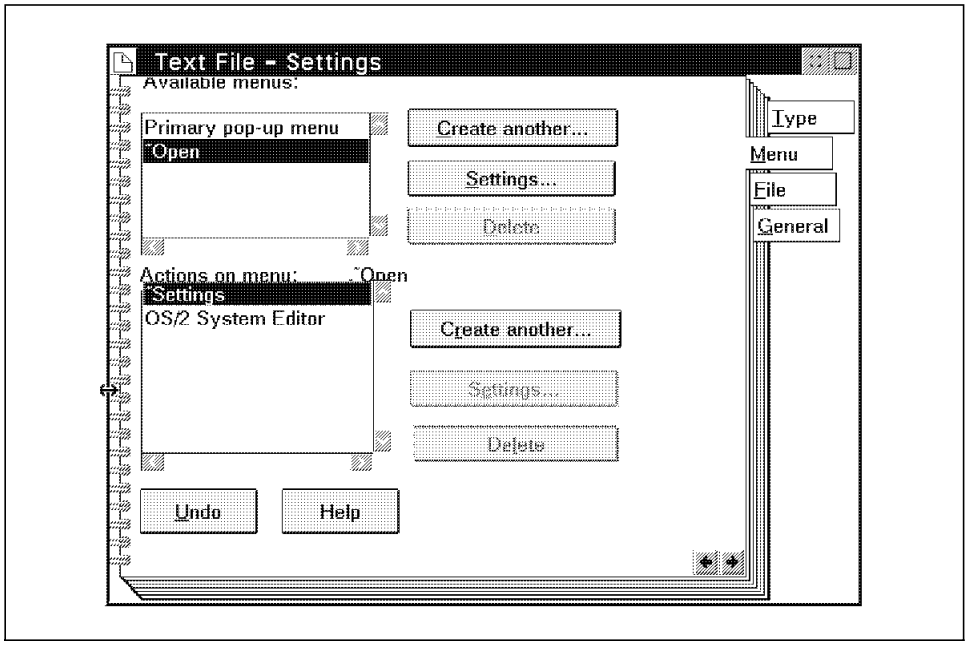


Figure 192. Menu Settings for a Data File

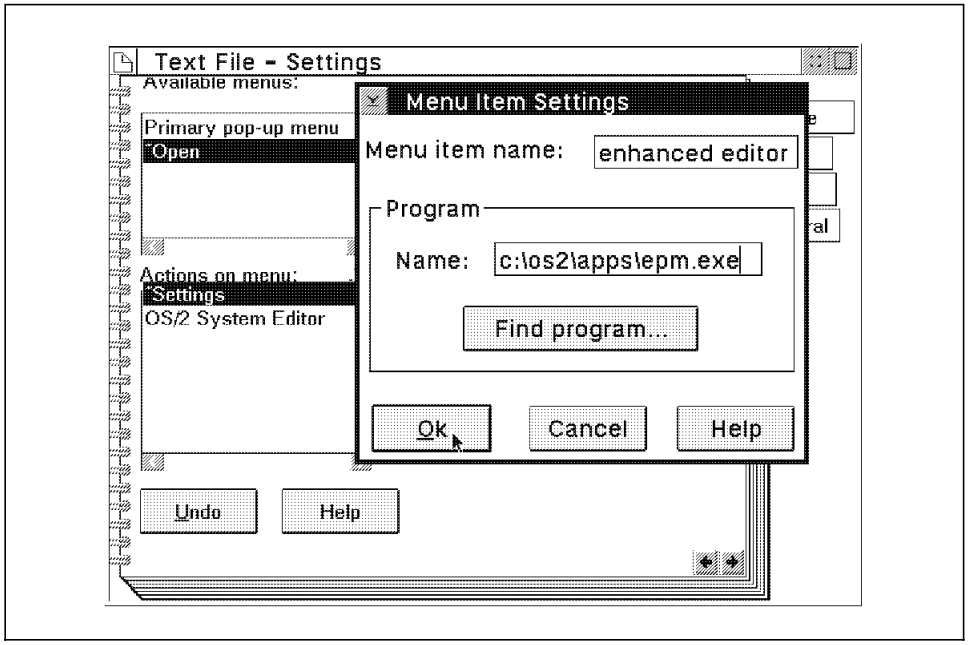


Figure 193. Menu Item Settings for the Enhanced Editor

The data file object is now associated with the enhanced editor. To make the enhanced editor the default we have to perform one more step. Making sure that the Open menu is still selected in the upper listbox click on the upper Settings button. A dialog will be displayed (See Figure 194 on page 238) with a drop down listbox allowing you to select the default option, that is the action that will be taken if you double click on the data file.

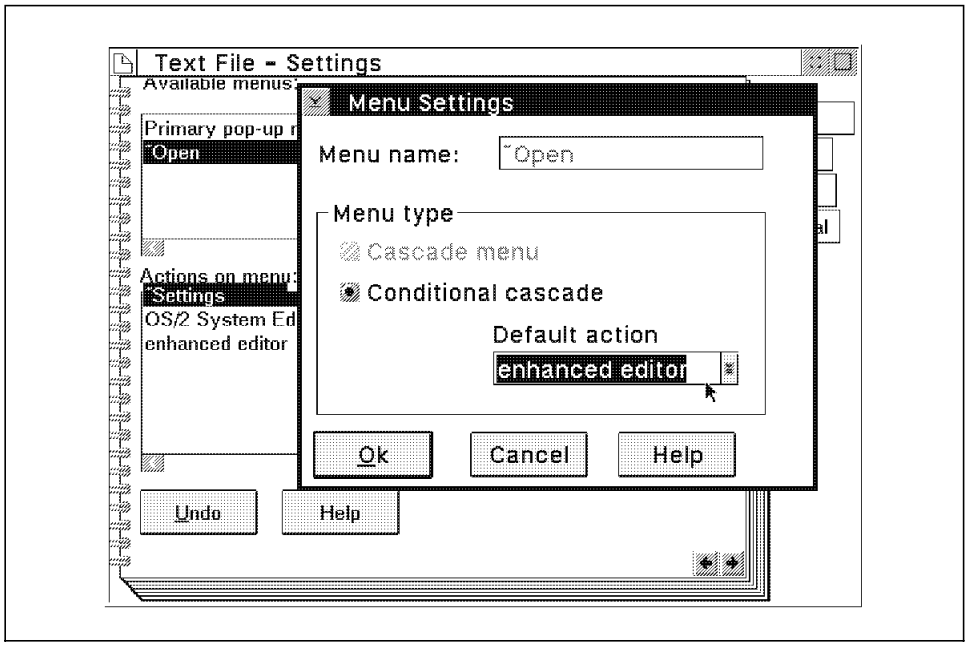


Figure 194. Setting the Default Open Menu Choice

Now if you close everything up and double click on the data file object you should see the EPM editor open with that data file.

NOTE

This process only works for programs that can take file names as command line arguments. If you happen to run across one that does not (and most do) then you will probably see the program open when you double click on the data file but the data will not be loaded. If this happens you will have to load the data into the program manually.

B.3 Associating Groups of Files

It may have struck you in the previous section that it could be a little tedious going to every one of your data files and setting up your favorite program as the default for each file. This is why the Workplace Shell designers have included a way to tell a program that it is associated with a whole group of data files and furthermore that any new data files fitting the criteria are associated with this program too.

B.3.1 Creating Group Associations

The first step in setting up a global association is to select the program you wish to associate and open its settings view to the Association page. We will use the enhanced editor as an example (See Figure 195).

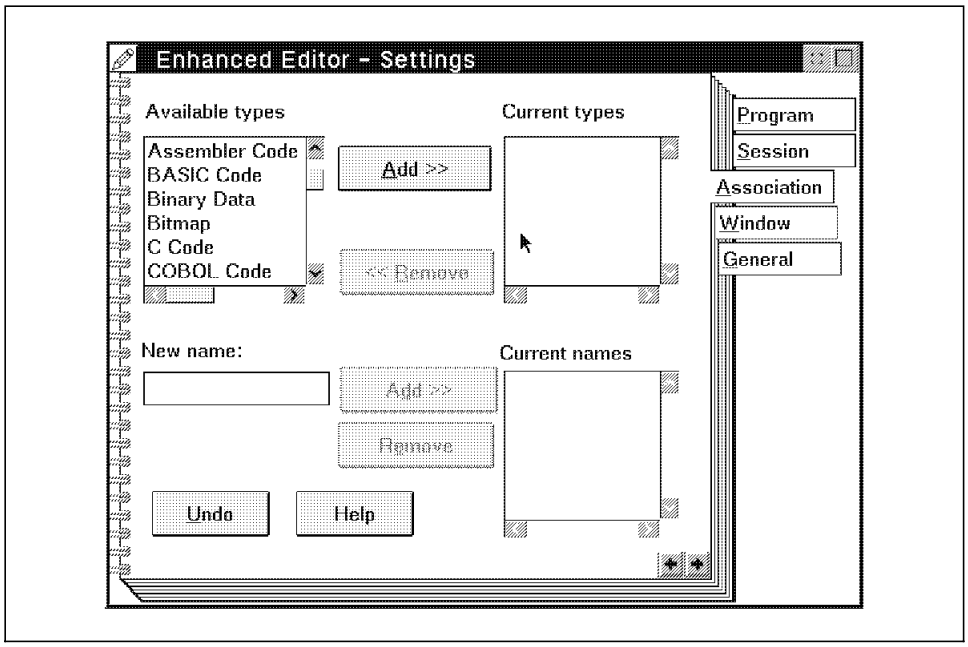


Figure 195. Association Settings for the Enhanced Editor

Notice here that you can define associations using two different classification schemes. The first is based on a data type which can be associated with a data file by some programs (such as the OS/2 System Editor which will not let you save a file unless it has an associated type). It is also possible to specify file types by using the Type page in the settings for that file. Data files may also be of more than one type. A good type to associate the enhanced editor with would be plain text since this is exactly the sort of file it

was set up to handle. To associate it with EPM simply select Plain Text in the listbox to the left and click on the Add>> button. Plain Text should now be in the right hand list box (See Figure 196 on page 240). To delete an existing association you can use the <<Remove button.

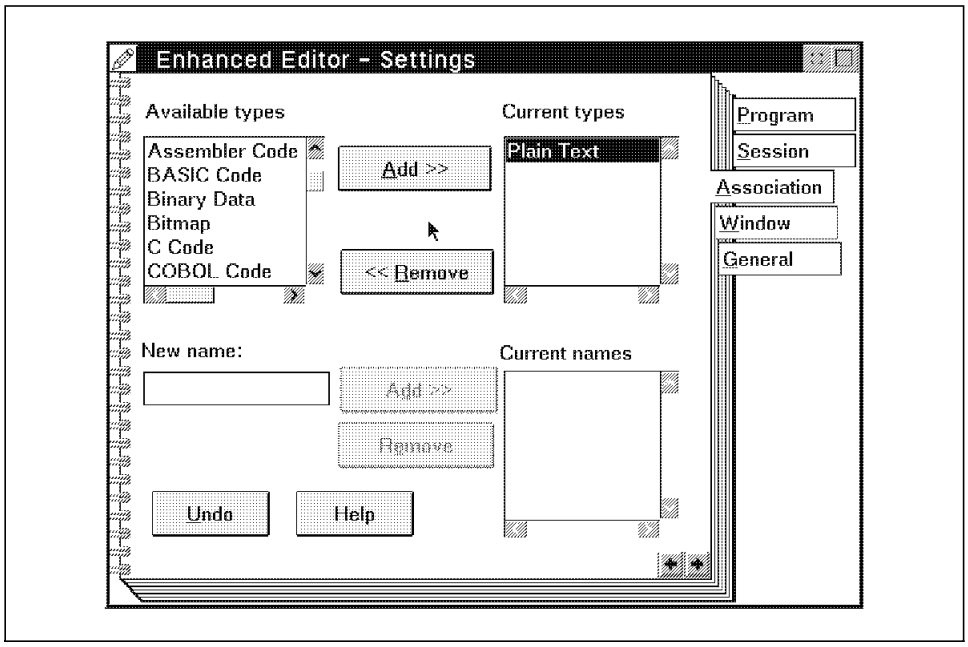


Figure 196. Adding a Type Association

The second form of association utilizes the entryfield and listbox at the bottom of the associations page. In the entry field you can enter file masks for files to be associated with the program. In our example we will associate EPM with the following file masks:

- *.TXT
- *.DOC
- README
- READ.ME
- CONFIG.* (for config.sys and variations)

To include these masks first type them into the entry field one at a time and then click on the Add>> button. The file masks should appear in the listbox to the right (See Figure 197 on page 241).

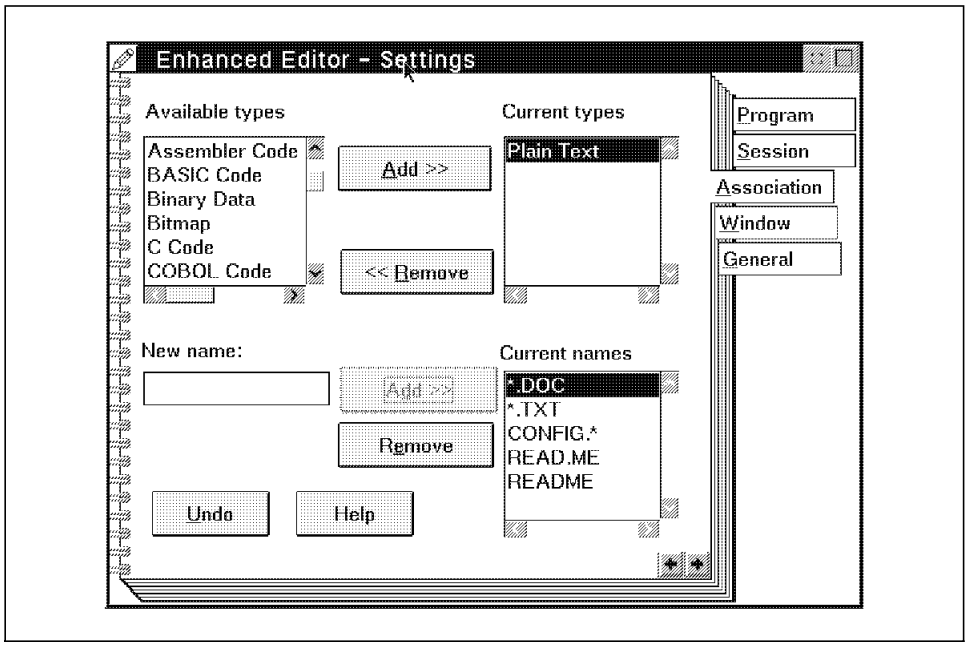


Figure 197. Adding a File Mask Association

After you have entered these associations any file matching either the types you specified in the upper listbox or the file mask you specified in the lower listbox will be associated with EPM (unless of course you modify the associations of the file itself by the method given in B.2, “Associating a Single File” on page 235).

B.3.2 Default Group Associations

The default behavior for group associations is a little more complex than with single files. The general rule is that the first association that is created is the default for that data file. If there is more than one program that has been associated with a given file type or file mask then the first program that was associated with that type or file mask is always the default for any data file, even if there is more than one type or file mask that matches.

To change the default association you must first disassociate all programs that could have an association with the file, associate the program you want as the default and then reassociate all of the other programs you also want on the data file's context menu.

Warning!

Do not try to fix global associations by manipulating an object's type and filename in such a way that it has no association. If you do this the object's settings view becomes the default and the only way you can change this is by using file associations as described in B.2, "Associating a Single File" on page 235.

B.4 Template Associations

Another way you can set up associations is by using a data file template with the correct associations already established. Files created using this template will then automatically have the same associations. There are two ways in which you can set up such a template:

- Changing an existing template
- Creating a new template

B.4.1 Changing an Existing Template

Changing the existing Data File template allows changes to the default data files that you create with it. The steps involved are exactly the same as the ones used in creating file associations described in B.2, “Associating a Single File” on page 235. For example if we set up associations as shown in Figure 198, each data file that we create using this template will have the same associations.

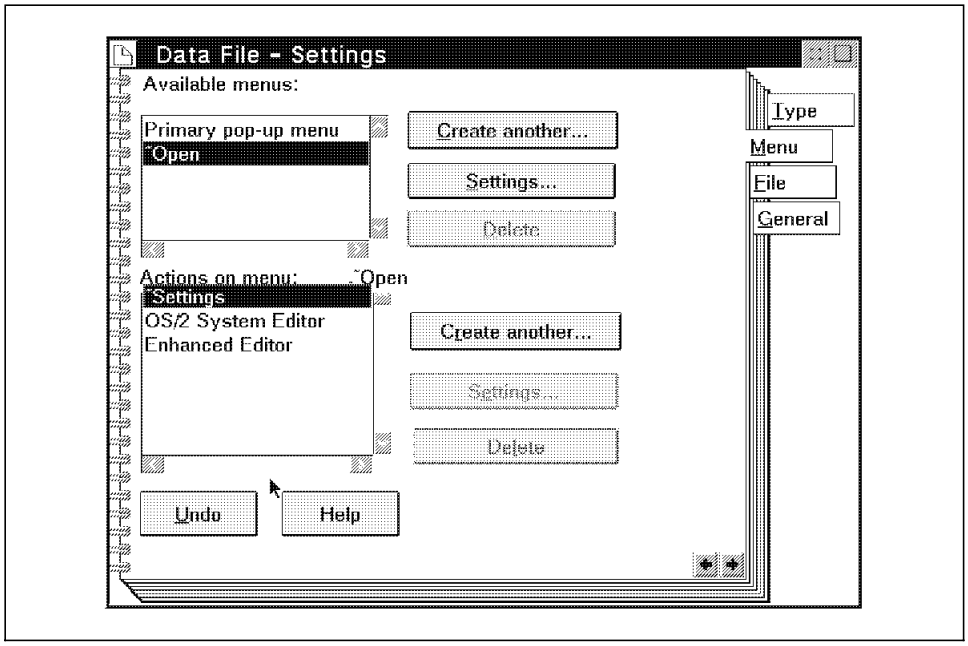


Figure 198. Altering the Associations for the Data File Template

B.4.2 Creating a New Template

To create a new template we must first set up a data file that has all of the properties we wish to duplicate each time a new object is created from it. Be aware that any contents that the file has will also be replicated along with associations and other settings. This enables us to set up things like form templates that contain skeletons to be filled in with data or REXX file templates that contain standard comment lines at the beginning. To create a template this way perform the following steps:

1. create a data file with any data you want to be included in objects created from the template
2. Set up associations for this file using the method described in B.2, "Associating a Single File" on page 235
3. Open the settings for the data file and go to the General page (See Figure 199). Near the bottom of the General page you will see a checkbox labelled Template. Select this box so that it is checked
4. Close the settings

Your data file has now become a template. To create new data files from this template simply drag using mouse button 2 from the template to where you want the new data file.

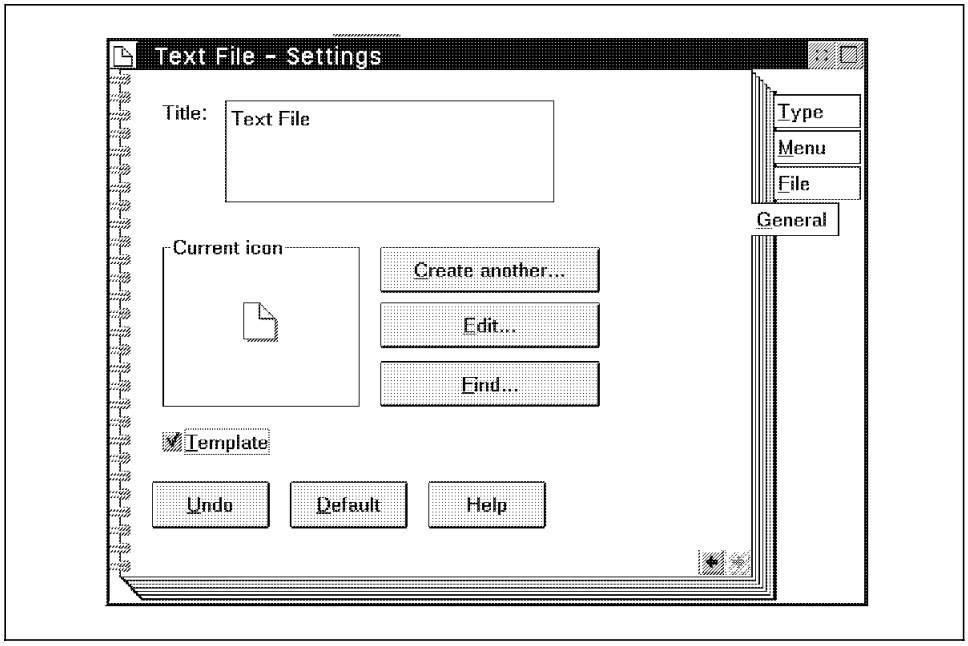


Figure 199. Menu Settings for a Data File

B.5 Using Associations

Now that we are able to create associations and templates we can look at ways to configure the Workplace Shell to better suit our work patterns. Up until the creation of OS/2 and even afterwards, the way people generally worked with applications was to decide what to work on, decide which application they needed, find the application, open the application, go to the file menu, find the data file and finally load it. With OS/2 and associations, however, a number of these steps can be removed leading to what is called a ′data centric′ way of working which means you can first decide which data to work on and then open the data object which in turn loads the appropriate application. This idea can lead to quite a number of different desktop organizations, two of which are described below.

B.5.1 Project Based Organization

To construct a project based organization you set up one folder for each project, subproject or task that you will be working on. Within each folder you place the data files (or shadows of the data files) that belong to each project, subproject or task.

The next step is to set up the correct associations between the data file objects in your folder and the applications you wish to use to manipulate them. You may also wish to place some data file templates in the folder so that you can create new data objects as necessary.

Now all you have to do to work on a particular project or task is to open the correct folder and double click on the particular item you wish to work with, the correct application will be loaded automatically. Alternatively you could choose to work with another associated application by using the options beside the open choice on the data file object′s pop-up menu.

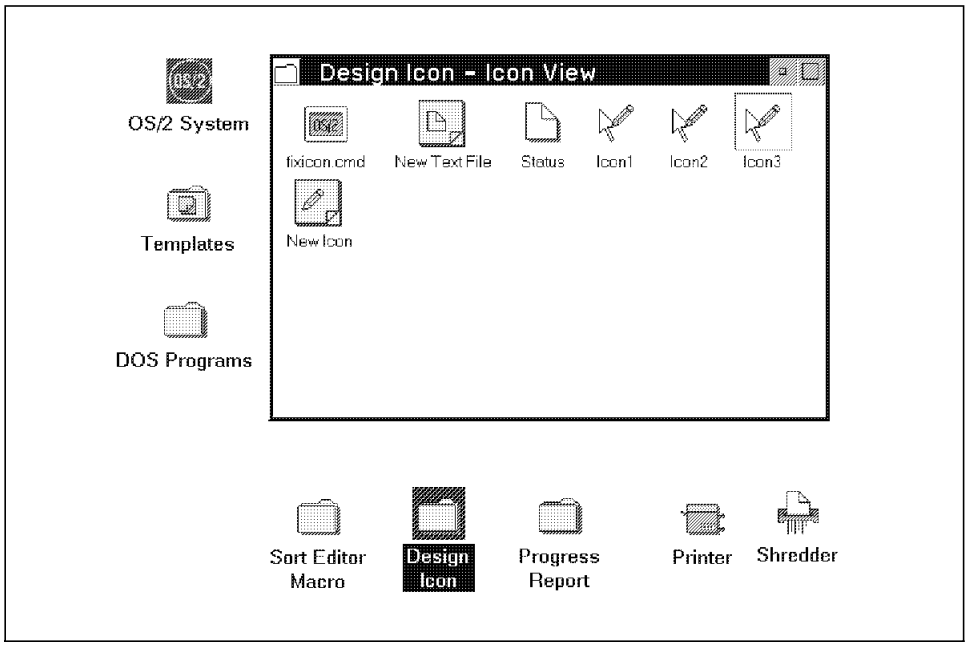


Figure 200. Example of a Project Based Organization

The example shown in Figure 200 shows a part of a desktop with a number of project folders, one of which contains different data file objects associated with a fictitious ″Design Icon″ project. Note that the different objects have taken on the icon of the default associated program even though they are all data files. This includes the OS/2 command file FIXICON.CMD which has taken on the icon of an OS/2 command window which is the default program for *.CMD data files.

B.5.2 Type Based Organization

Another way to organize a desktop is to use a type based organization scheme. In a type based organization you set up one folder for each type of data file you want to work with. In each folder you place a template for that sort of data file object so that you can create new objects easily without having to go hunting for the templates folder. The example shown in Figure 201 on page 247 shows two folders with different types of data files. Note that both the REXX command file folder and the icon file folder include the relevant templates. Although you cannot see it from the figure we have set the REXX data file template to include a REXX program skeleton for each new REXX data file object.

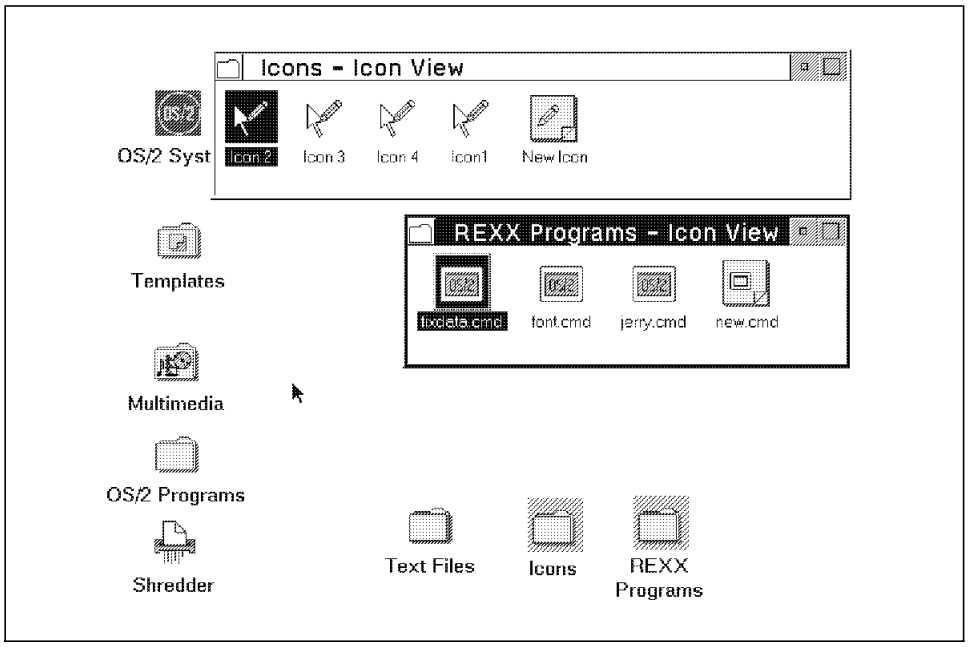


Figure 201. Example of a Type Based Organization

B.6 Work Area Folders and Associations

Work area Folders are one of the more under used features of the OS/2 Workplace Shell. To understand them we first need to look at the desktop. When you shut down your machine and close the desktop the Workplace Shell saves information about which programs were running. When you reboot your workstation these programs will restart automatically, provided you haven't modified your CONFIG.SYS to prevent this. Work area folders use exactly the same concept to restart programs at an individual folder level.

If you designate a particular folder as a Work area Folder then any open object within that folder will be closed when you close that folder. Furthermore when you re-open that folder all of the objects that were open when you closed the folder will be re-opened automatically. This includes data file objects that may be associated with the appropriate applications. Minimizing the folder's open view has a similar effect, that is, when you minimize the folder all of the object views that were opened from that folder will be minimized also.

When you combine Work area folders with one of the organization schemes described in B.5, “Using Associations” on page 245 you have a powerful way of setting up your shell to be more productive, particularly if you are using some form of a project based organization scheme. If you have set up the appropriate associations then when you have finished working on a project or task for the moment you simply close the work area folder that contains your project data files which in turn closes all of the applications associated with those files. When you want to work on that project or task again you double click on the Work area folder for your project and all of the data files that were opened when you last closed the folder will be opened just as if you had double clicked on them. This in turn will load all of the associated applications to work with those data files and load the data files into the applications. Thus with one double click you can be back nearly to where you were when you were last working in that folder.

B.6.1 Setting Up a Work Area Folder

To set up a Work area folder you simply change one setting of the folder that you wish to be a work area. Perform the following steps:

1. Open the folder’s settings view by clicking on the arrow to the right of the open option on the folder’s pop-up menu (See Figure 202 on page 249)
2. Click on the File tab to bring up the File page and check the Work Area checkbox (see Figure 203 on page 249)
3. Close the settings and your folder is now a work area folder.

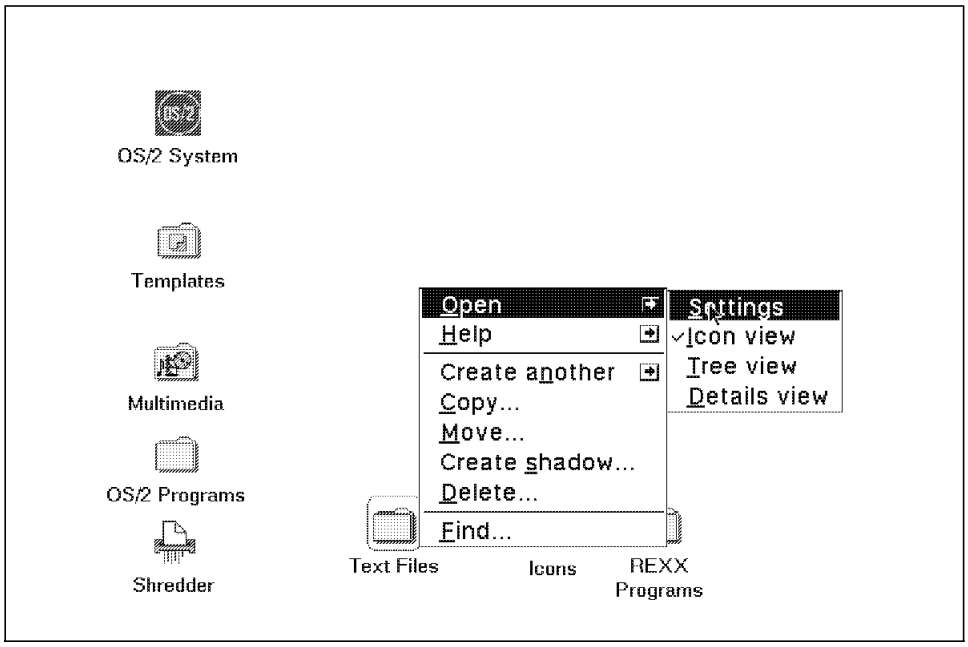


Figure 202. Opening Settings for a Folder

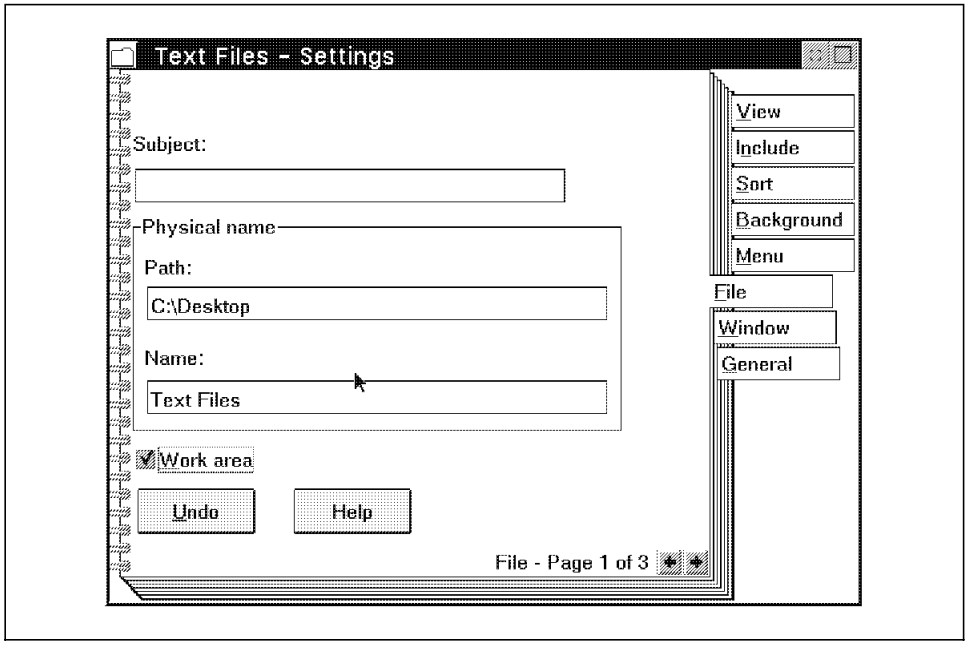


Figure 203. Making a Folder a Work Area

For an an example of the results of opening a work area folder see Figure 204 on page 250 showing several copies of the system editor opened automatically when the text files work area folder was opened.

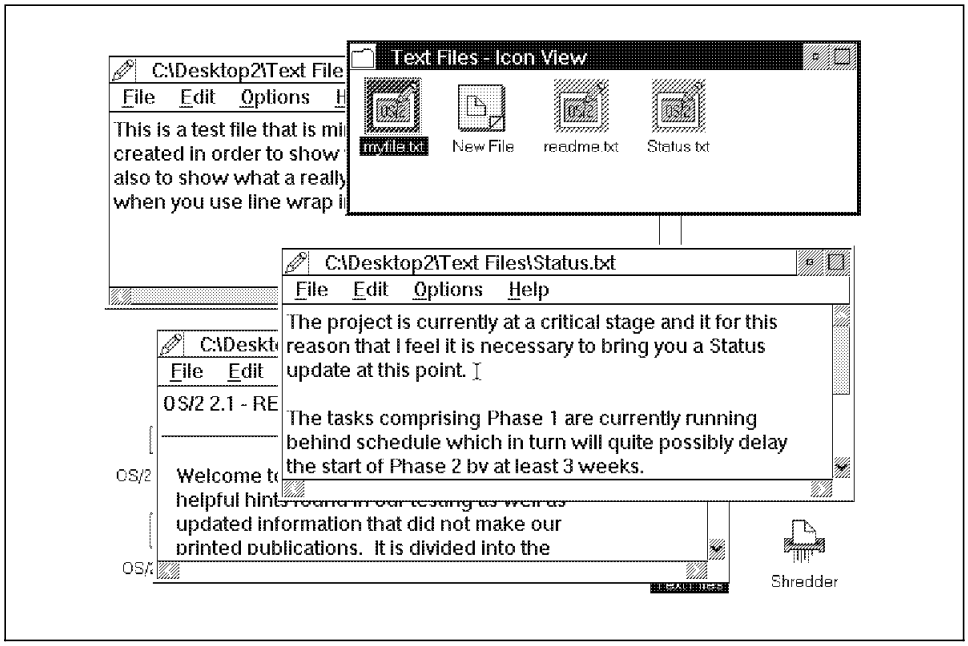


Figure 204. Results of Opening a Work Area Folder

Appendix C. RC Syntax Diagrams

We have included this appendix to give you a quick reference to the syntax of the OS/2 Resource File.

Throughout this section, syntax is described using the structure defined below:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ▸▸— symbol indicates the beginning of a statement.

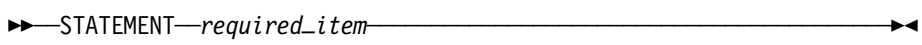
  The —▸ symbol indicates that the statement syntax is continued on the next line.

  The ▸— symbol indicates that a statement is continued from the previous line.

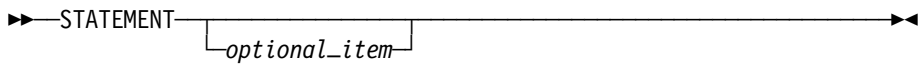
  The —▸◂ symbol indicates the end of a statement.

  Diagrams of syntactical units other than complete statements start with the ▸— symbol and end with the —▸ symbol.

- Required items appear on the horizontal line (the main path).

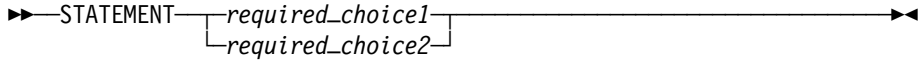


- Optional items appear below the main path.

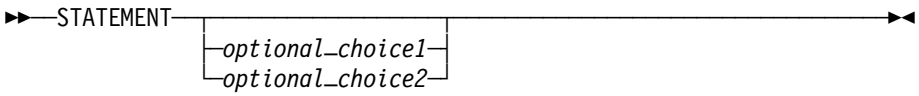


- If you can choose from two or more items, they appear vertically, in a stack.

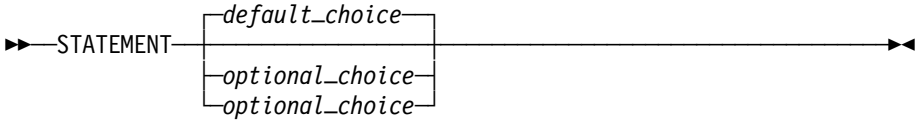
  If you *must* choose one of the items, one item of the stack appears on the main path.



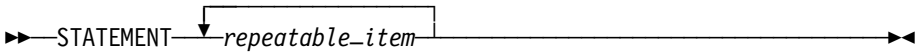
  If choosing one of the items is optional, the entire stack appears below the main path.



- If one of the items is the default, it will appear above the main path and the remaining choices will be shown below.



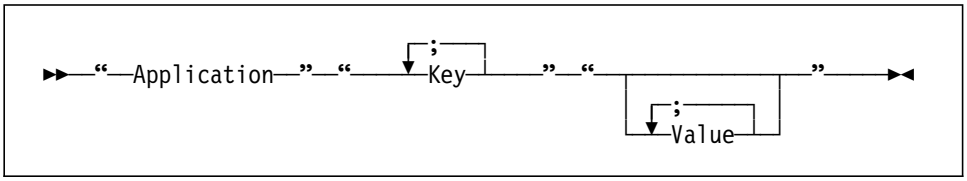
- An arrow returning to the left above the main line indicates an item that can be repeated.



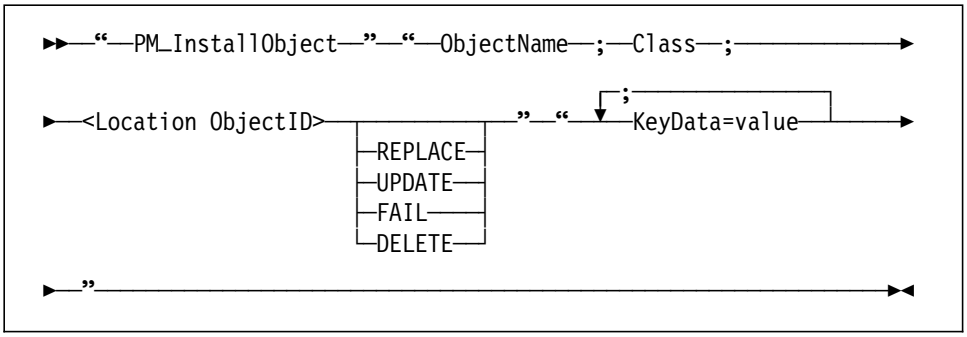
  A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Keywords appear in uppercase (for example, PARM1). They must be spelled exactly as shown. Variables appear in all lowercase letters (for example, *parmx*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, you must enter them as part of the syntax.

C.1 Keyword Instructions



C.2 PM_InstallObject Keyword Instructions



Appendix D. Deskman/2

Due to the fact Deskman/2**
(DM/2)** from Development Technologies Inc deals specifically with managing the Workplace Shell, we felt we should do an evaluation of this product. This appendix will deal with what we discovered during our testing of Deskman/2. This is in no way a complete users guide for this product but simply a report of what we found with the testing that we did with the product.

The Deskman/2 (DM/2) product will actually put four components on the hard drive of your system during installation and will create four objects on your desktop:

- DeskMan/2
- DM/2 Image
- VUEMan/2**
- DeskMan/2 User's Guide

We will discuss some of what each of the components does, and some of the results we received when we tested the functions in each section. The version of DeskMan/2 we used is 1.20, a different version may have different results from what we received and discuss in this book.

D.1 DeskMan/2

The DeskMan/2 object contains a lot of the main functions of DM/2 and allows you to Save objects from your desktop, either one by one, in groups or the entire desktop at once. DM/2 supports the OS/2 drag and drop function so you can drag individual objects form your desk top to the DeskMan/2 object and they will be saved.

If you don't want to use the drag and drop function you can save or restore the desktop by using the DM/2 main menu. This menu can be accessed in 2 ways:

- Click mouse button 2 on the DeskMan/2 object
- Bring up the DM/2 window and then click mouse button 1 on the top left corner of the DM/2 window, on the little DM/2 icon.

An example of accessing the main DM/2 menu from the DM/2 window is in Figure 205 on page 256

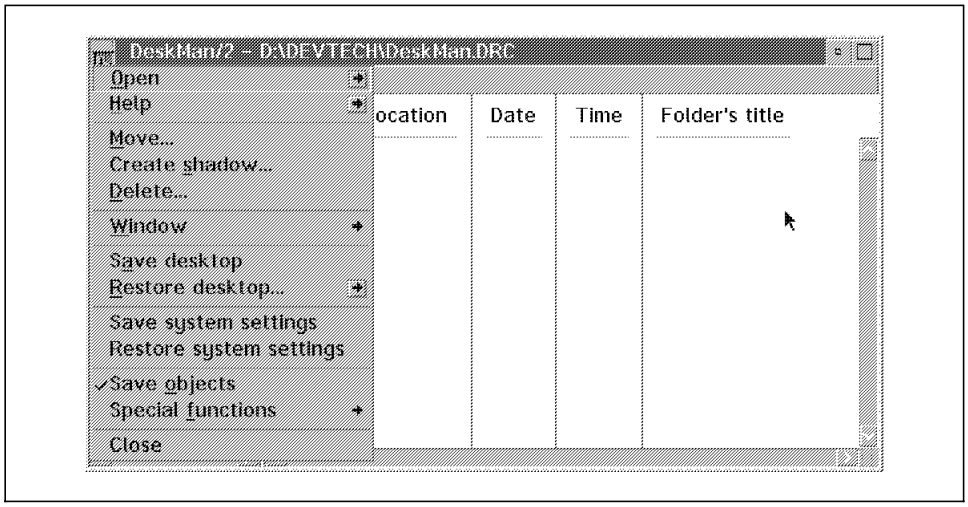


Figure 205. DM/2 Window and Menu

There are several options in the DM/2 menu which are normal for most objects on an OS/2 Desktop, such as Open, Help, Move... etc. There are also several options that allow you to make use of the functions in DM/2. A list of these functions and a brief description follows:

- **Save desktop** - will save your entire desktop into a file named Deskman.drc, which is the default. You can not change the name of the file you save to, so if you wish to have more then one desktop save file on your system you have to rename the file after you have completed the save operation.
- **Restore desktop...** - has 3 options you can chose from before it then brings up a window that allows you to chose which file to restore.
  - **Update if exists** - will update any object that already exists on your desktop, with any new information that the saved object contains.
  - **Replace if exists** - will replace any object that already exists on your desktop, with the one from the saved file.
  - **Skip if exists** - will skip any object that already exists on your desktop.
- **Save system settings** - will save your system settings like colors, border size, cursor blink rate.... etc, into a file called Deskman.IRC in the

DEVTECH directory. The file name that DM/2 saves the settings into cannot be changed.

- **Restore system settings** - will restore the system settings from the Deskman.IRC file, provided you have saved the settings prior to doing the restore.
- **Save objects** - this is the default for the DM/2 and will stay in effect until you make a selection from the Special functions. You can use this selection to turn off any Special function selection you might have used.
- **Special functions** - this option will give you access to a second menu which contains 7 selections. Choosing any one of these options will cause the DeskMan/2 icon to switch to something that tries to illustrate what function DeskMan/2 is set for.
  - **Destroy object on drop** - DeskMan/2 icon will now delete any object dropped on it.(similar to the OS/2 shredder)
  - **Query object settings on drop** - if an object is dropped on the DeskMan/2 icon a window will be opened which contains the settings like ICONFILE= and PROGTYPE=, similar to Figure 208 on page 260. it will also contain a *Gen REXX* button which will generate the REXX code needed to recreate the object.
  - **Assign object ID on drop** - if an object is dropped on the DeskMan/2 icon a window will be opened which will have the existing object ID in it. If there is no existing object ID, then it will display <?>. You can use this window to change the object ID if you wish.
  - **Open object on drop** - has 2 options
    - **Default** - opens any object or objects dropped on it
    - **Settings** - opens the settings of any object or objects dropped on it
  - **Assign ICON on drop** - will allow you to drag an existing ICON, or one that you create, onto DM/2. The ICON will be saved and then you will be prompted to drag the object or objects you wish to use the icon for onto DM/2. When you do this the objects will now have the new ICON in place of what ever icon they had prior to doing this procedure.
  - **Change object style on drop** - when an object is dropped on DM/2 a DM/2 Change object style window will appear which has various options in it like Copy, Move, Shadow..... You can select whether or not you wish to have each of the options active or not and also whether to make the change permanent. If you chose not to make it

permanent the next time the WorkPlace Shell is reloaded the object returns to it's previous settings.

– **Force WPS to save object settings on drop** - will cause the WPS to save the settings of any object that is dropped on DM/2.

If you only want to save selected objects or don't want to use the Save on the menu, you can use the drag and drop feature. This allows you to select one or more objects, from your desktop, and drag them to the DM/2 window or object. This will save the selected object and any objects that are contained within it, providing the DM/2 ICON is set for *Save objects*. The example in Figure 206 was done by dragging the OS/2 Multi Media object onto DM/2.

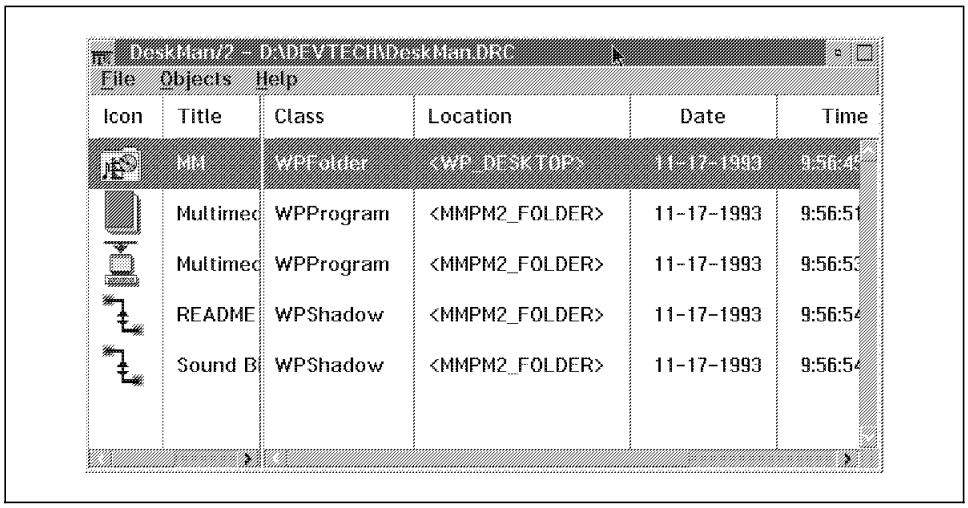


Figure 206. Saved Multi Media Folder

Now that you have some objects saved in the DM/2 window you can manipulate the objects if you wish. The way to do this is to bring up the menu, that is in Figure 207 on page 259, which contains the options that allow you to work with the selected object. There are two ways to get this menu up:

- select the object you want to work with, from the list in the DM/2 window, and then click mouse button 2 on the object.
- click either mouse button on the *Objects* option on the DM/2 window's action bar.

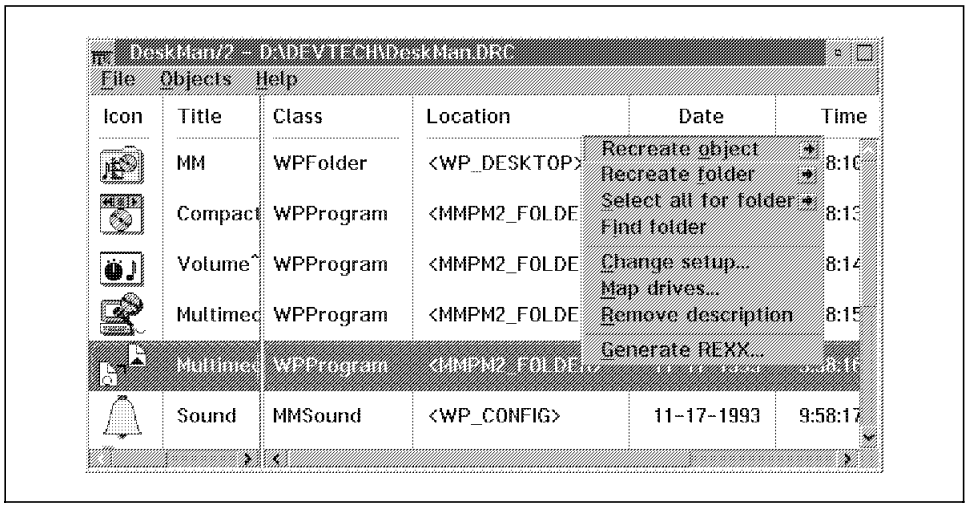


Figure 207. Menu to Work With Object

You can use this menu to manipulate the definition of any object or folder you have saved. The following section will give you a brief description of the major options and what they can do for you:

- **Recreate object** - will recreate the selected object onto you desktop using one of the 3 options which are the same as the options available in *Restore Desktop* from the main menu.
- **Recreate folder** - will recreate the selected folder to your desktop and will recreate the subfolders as well depending on which one of it's 2 options you set:
  - This folder only
  - include subfolders
- **Map Drives** - will allow you to take a file that contains the saved definitions from one system and use it on another or move applications from one drive to another on the same system. A couple of examples of this would be:
  1. You save the definitions from a system that has LAN, CM/2, DB2/2 and several other applications installed on it's "D" partition. You then restore the definition file to a system that has the same applications installed on it but they reside on the "E" partition. You can select the applications from the definition list and then select the Map Drives option and set the *from* and *to* drive letters to indicate the correct drives. Also you can chose the *including ICON files* and the *including data files* buttons.

2. You decide to move some applications or objects on your system from one partition to another. You can select the applications from the definition list and then select Map Drives and make the appropriate selections to change the drive assignments in all of the selected applications or objects.

- **Remove description** - will remove any selected objects from the description file and they will not be restore during the next restore operation.
- **Change setup...** - will bring up the Change setup window which gives you the capability of making changes to the setup of an object. One example of this would be to change the icon of an object. Then the next time you do any restore the object will have the new icon. The screen in Figure 208 is an example of what the Change setup panel will give you access to. There are two ways to open this panel:

  1. Select Change setup... from the menu after
  2. Double click mouse button 1 on the object on the DM/2 screen which you want to work with.

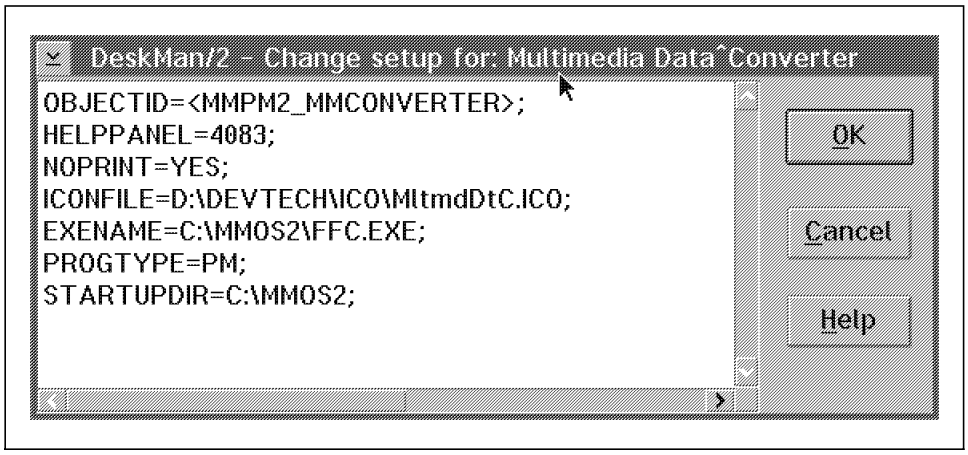


Figure 208. Change Setup

- **Generate REXX** - this option will bring up the *Select REXX program* window where you can name the REXX CMD that will be generated. The CMD will contain the REXX code that is required to recreate the selected object. This code can then be used in a REXX program to create a duplicate of the object on other systems.

D.2 DM/2 Image

This object can be used to switch from one desktop to another while the system is up and running. One use for this might be if you have more then one person using a single system and both people wish to have a customized desktop. Using the DM/2 Image will allow the users to switch from one desktop to another.

If you double click on the DM/2 Image object with mouse button 2 you will get a DM/2 Image panel like the one in Figure 209.

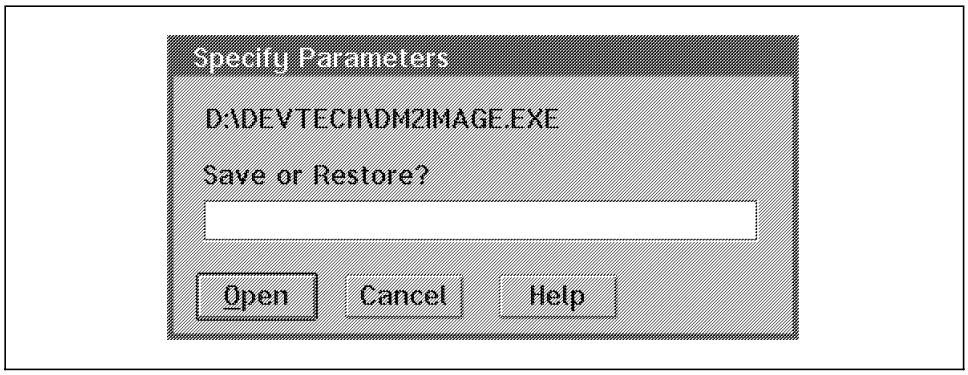


Figure 209. DM/2 Image Panel

The only things you are allowed to do in this panel are specify Save or Restore and which file to save to or restore from. An example of both these would be:

- Save Desktop1.rep - will cause DM/2 Image to save the current desktop in a file called Desktop1.rep in the directory that DM/2 Image is loaded in. It will also run a verify on the save.
- Restore Desktop.rep - will cause the desktop that was saved in the Desktop1.rep file to be restored onto the system. This will eliminate what ever desktop is on the system, at the time this command is run, and replace it with the one from the file.

After typing in the command you wish to run you select the Open button and the system will execute the command.

We found that the save function took approximately one and a half minutes depending on the number of objects on the desktop. The Restore would take approximately 2 minutes or so to complete depending on the number of objects on the desktop that was being restored.

If you have an application running at the time you save the desktop, the application will be brought up when you restore that desktop. Also some applications that are running on the desktop at the time you do a restore will be running on the new desktop when it is restored. How ever there are some things that do not do this. The system applications such as the System Clock, Color Palette, Scheme Palette and Font Palette will not be restarted. These are all WorkPlace Shell programs that are only DLLs with no EXE file associated with them.

D.3 VUEMan/2

When you select the VUEMan/2 object for the first time it will start the default VUEMan/2 panel which will have four sections in it like the one in Figure 210.

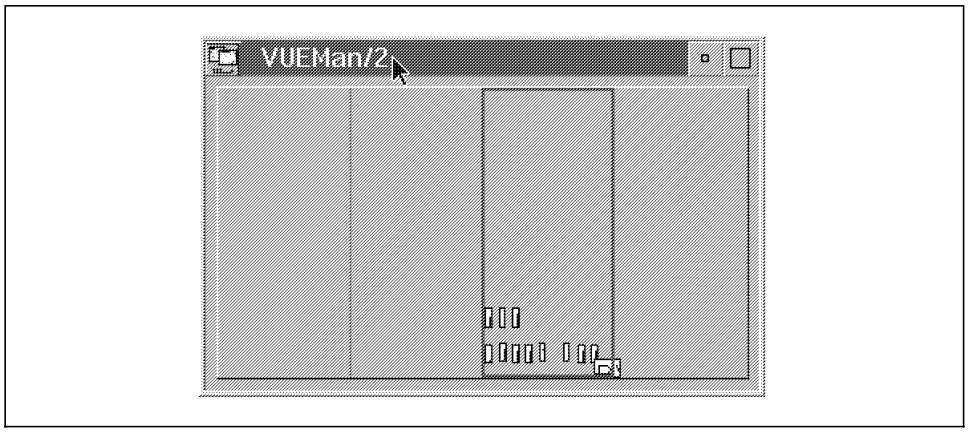


Figure 210. Default VUEMan/2 Panel

You can use VUEMan/2 to help organize your desktop by placing the different applications you are running in separate sections, this will clean up your desktop and make it less cluttered. The way to do this is:

1. Start VUEMan/2
   - If you already have applications running they should show up in the section of the VUEMan/2 window that is highlighted.
   - If you do not have any applications running start the ones you wish to use.
2. Select the application from the active VUEMan/2 section, by clicking mouse button 2 on it and holding it down while you drag it to the new section.

3. When you release the button the application will disappear from the actual OS/2 Desktop as well as from the original VUEMan/2 section and it will now be in the new section.
4. Continue doing this until all of the applications you wish to move have been moved.

After setting up the VUEMan/2 you can select the application you wish to use by double clicking on the appropriate section and the application will appear on the screen. You will also have your normal background, and objects on your desktop in each of the sections. The only thing that will change from one section to the other should be the actual application that is in the foreground and displayed on your screen. You can place several applications in the same section or chose to have only one in each section.

If the default of 4 sections is not enough for you it is possible to expand the VUEMan/2 window to a maximum of 81 sections, in a 9x9 grid. The way to do this is:

1. Start VUEMan/2
2. Click mouse button 2 on any part of the background of one of the sections
3. Select options from the menu that comes up
4. Set the Desktop Width and/or Height to the number of sections in the grid of your VUEMan/2

We decided to increase the Height of our panel from 1 to 2 and our VUEMan/2 then had 8 sections in it so we were able to place each of our running applications into their own section. The screen in Figure 211 on page 264 displays our VUEMan/2 after we configured it.

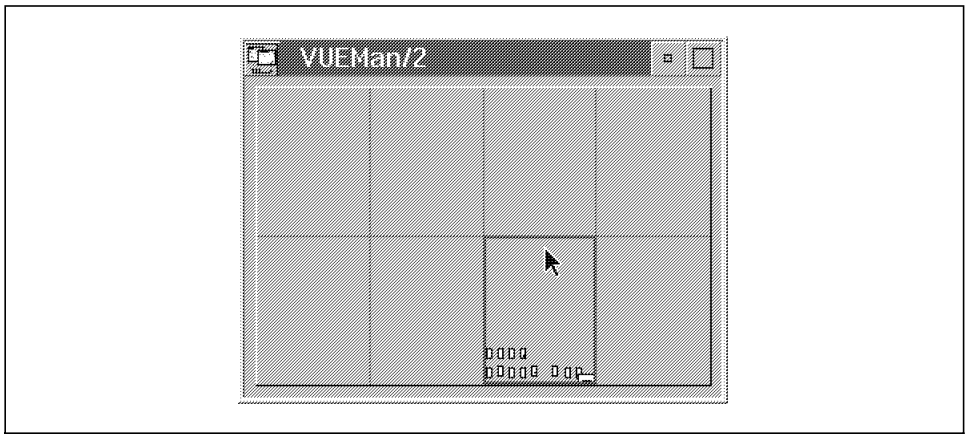


Figure 211. Expanded VUEMan/2 Panel

After you have expanded the panel to what ever size you want and placed your applications into the sections you want, you can save the *Layout* of your desk top by returning to the VUEMan/2 menu and selecting Layout. When the Layout window comes up you chose the *Add Current Layout* button and then give your Layout a name when prompted.

You can save several different desktop layouts which will allow you to have different combinations of applications available at different times by changing the layout you are using. When you select a layout the applications will be placed in the sections you saved them in, provided they were running prior to starting the layout. If you did not have some of them running at the time you changed the layout then you simply have to start them and then go into the VUEMan/2 and select the correct layout again and they will be moved to where they were saved. You can also drag and drop them to the correct spot if this is what you would like to do. VUEMan/2 is a good way to get away from having a cluttered desktop due to multiple applications running at the same time and having to minimize and maximize applications.

The next 2 Figures will show you an example of a cluttered desktop and then what it looks like after setting up VUEMan/2.

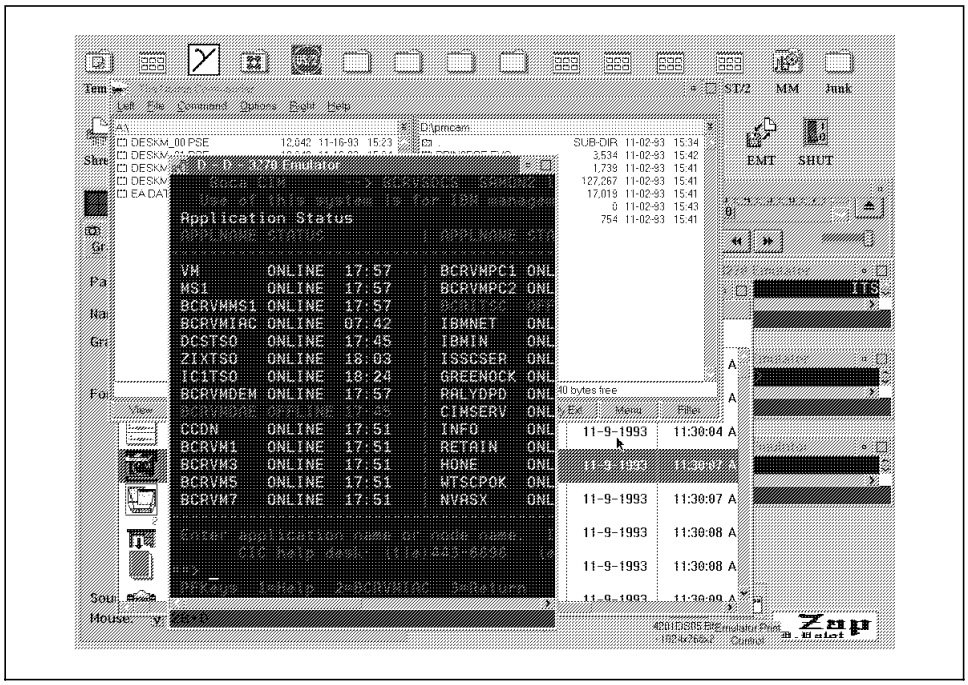


Figure 212. Cluttered Desktop

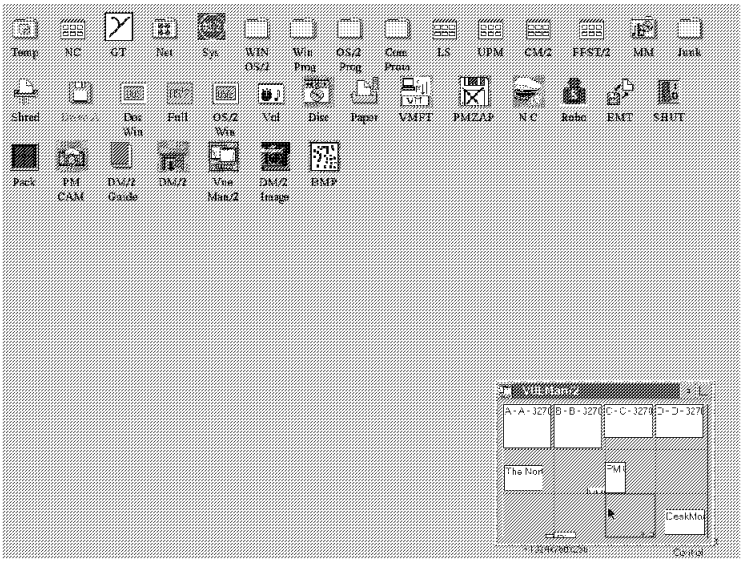


Figure 213. Cleaned up Desktop

click mouse button 2 on the background of any of the sections in the VUEMan/2 panel.

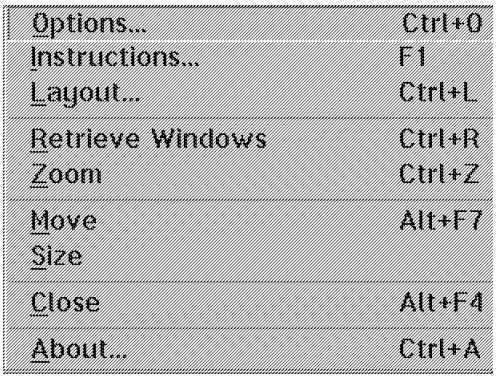


Figure 214. VUEMan/2 Main Menu

If you select *options* you will get the Options menu displayed on your system.

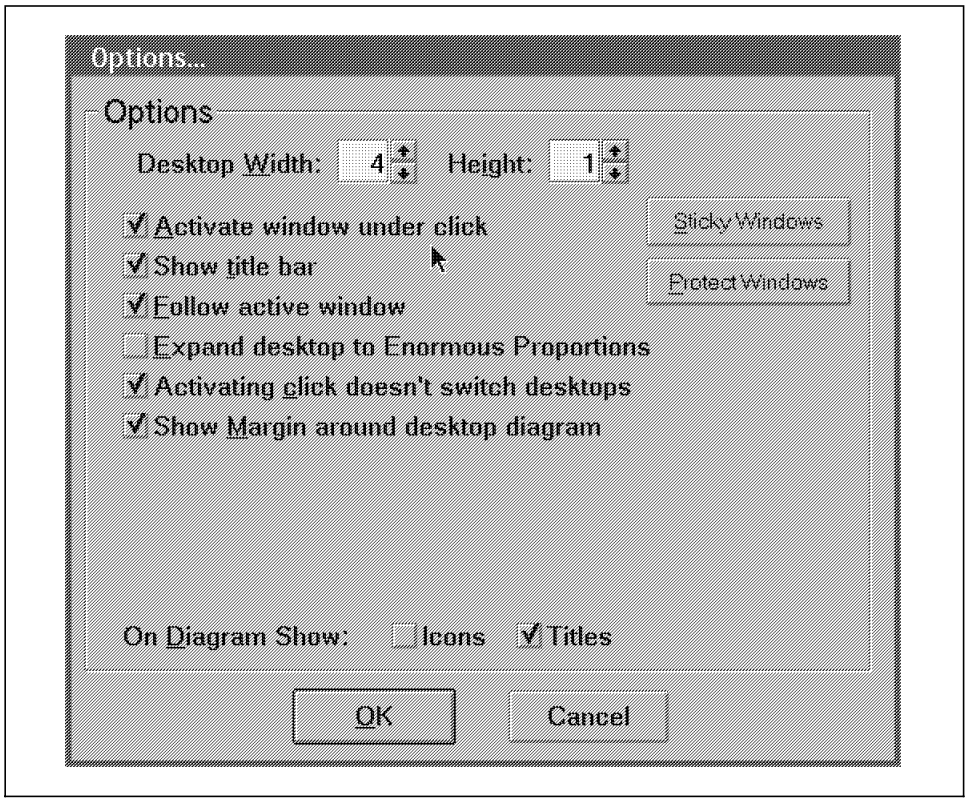


Figure 215. VUEMan/2 Option Menu

D.4 DeskMan/2 User's Guide

This object is the online User's Guide for the DeskMan/2 product. Selecting this object will take you into the guide and you can do the regular OS/2 searches, hyper text and the normal functions that are available in the OS/2 Help and View functions.

D.5 DM/2 Command Line Interface

DM/2 comes with some Command files that you can use to interface with DM/2 from an OS/2 Command line. These command files are:

1. Installation and configuration command files
2. Operation command files
3. Object creation command files

DM/2 is also capable of creating other command files.

D.5.1 Installation and Configuration Command Files

There are two Installation and configuration command files that come with DM/2:

1. DESKMANR.CMD
2. DESKMANS.CMD

These CMD files can be used, from a command line, to install and configure DM/2. You must use the DESKMAN.CF! and DESKMAN.CFG configuration files with these CMD files. The configuration files must be edited prior to using them. Doing the install this way would allow you to specify which features and functions get installed.

D.5.2 Operation Command Files.

There are several command-line operation files.

1. CLENUP.CMD - remove excess files from the DM/2 storage directories, these are not referenced files
2. REMOVE.cmd - remove DM/2 from the system
3. PERFDO.CMD - destroy an object or objects on the desktop
4. PERFSO.CMD - save an object or objects from the desktop
5. PERFRO.CMD - restore an object or objects to the desktop
6. PERFSD.CMD - save a desktop′s objects
7. PERFRD.CMD - restore a desktop′s objects
8. PERFSS.CMD - save a systems settings
9. PERFRS.CMD - restore a systems settings

These files do not get installed during a normal DM/2 installation, but have to be copied from the DM/2 disks to the hard drive.

D.5.3 Object Creation Command Files

These are the files that you create when you use the *Generate REXX* option for any object on your desktop. These generated REXX programs will allow you to restore an object or objects to a desktop with out having DM/2 installed on the systems you are restoring the objects to. This is of course provided you have REXX installed on the target system.

Appendix E. Alternate Shells

In the six years of OS/2, IBM has shipped three different shells for OS/2. OS/2 1.0 had a character shell called the Program Selector. The shell for versions 1.1, 1.2, and 1.3 were actually a suite of Presentation Manager (PM)applications for program launching, spooler management, file management, and system control. OS/2 2.0 brought us the Workplace Shell, which was the first implementation of Common User Access (CUA)*. This was the shell for current and future versions of OS/2.

E.1 Why Use a Different Shell?

Using a different OS/2 shell is not for everybody, however some applications could use a different shell as they do not require the full function of the PMSHELL and all the overhead that goes with it. Some examples of these would be:

- bulletin board systems
- specialized servers such as CID servers
- embedded systems
- UNIX alternatives

Another use of an alternate shell could be an administrator wanting to limit the access users have to their OS/2 systems, such as not allowing them to open a command prompt. Because the Workplace Shell is based on object-oriented programming, it is possible to invoke existing methods or subclass Workplace Shell objects and change their behavior.

A logical first step is to password-protect folders. Subclass the WPFileSystem class, override the _wpOpen method, and present a password dialog box before invoking _wpOpen in the parent. The International Technical Support Center (ITSC) Red Book GG24-3774-00, OS/2 Version 2.0 Volume 4:Application Development, p. 107, has some sample code for this.

Some of the other options available, for alternative or customized shells, have been discussed in the previous sections of this book. The next sections will discuss using a couple of alternative shells that have already been built and are included on the diskette that was shipped with this book.

E.2 TShell

You will find a TShell.ZIP file on the diskette that was included with this book. TShell is a character-based, protected-mode shell for OS/2 2.x which makes OS/2 2.x look a lot like old OS/2 1.0. The procedure for installing the TShell is very simple:

1. Install a text editor such as EDLIN or other NON PM editor you like because you need an editor to be able to remove the TShell. Seeing as the TShell does not support PM applications you must use a a character based editor.
2. Unpack the TShell.ZIP file using PKUNZIP.
3. Copy TSHELL.EXE into the root directory of the boot drive.
4. Edit CONFIG.SYS and change the PROTSHELL setting to:

   PROTSHELL=TSHELL.EXE
5. Copy PGMSHELL.EXE into your PATH.
6. Reboot the system.

When the system comes up you will find that you are now in the TShell Desktop and you no longer have objects on your desktop. In place of the objects you will see the main screen which will contain 2 boxes called:

- Start Group - Contains the names of things you can start.
- Running Group - Will be blank but will contain the names of any thing you start.

You will also notice that there is no mouse support however, as always, you can still use alt-esc to switch around running sessions and control-esc to bring you back to the main screen.

TShell will run DOS and Windows programs if the machine is configured to run these prior to changing to the TShell. There is also a ″Shutdown System″ selection on the Start Group which will shut the system down, however you have to manually shut down anything that is in the Running Group before the Shutdown will work.

One of the big advantages to the TShell is it use less resources to run then the normal shell. It can also boot very quickly depending on what drivers you load with it

There are several restrictions that exist with the TShell. Reference the following list to find out what the restrictions are:

- Like OS/2 1.0, there is no Presentation Manager, therefore OS/2 PM applications will not work and every TShell session will be a full-screen session.
- Don't attempt to run the TShell from a command line on a PM system as the system will crash TShell is meant for the PROTSHELL statement in CONFIG.SYS.
- TShell does not have a spooler in it. However this does not mean you cannot print while running the TShell. It does mean you should only send one print job at a time and you must have the printer defined and running prior to making the switch to the TShell.
- The command START /DOS does not work right. Instead of starting a DOS session it will actually start another OS/2 session.
- The RexxUtils will not work.
- At this time the IBM LAN Requestor does not work.

E.2.1 Programming the Start Group

TShell's list of startable programs can be modified by writing a script in REXX and invoking it via the utility PGMSHELL.EXE. PGMSHELL collects data from REXX and passes it to TShell. If you never use PGMSHELL, TShell will provide a list of items to start. Programming the Start Group will allow you to put any application you will be using into the Start Group. This way you do not need to have all of your applications running at the same time unless you want them all running.

Here is a sample REXX program that modifies the TShell start list:

```
/* rexx program to modify tshell */
 parse source . . szRexxFileName .
 select
      when 'CMD' = address() then do
        '@PGMSHELL' szRexxFileName
        if rc > 0 then do
          say 'PGMSHELL.EXE not found'
          return 2
        end
      end
      when 'PGMSHELL' = address() then do
        /* title text for the start list  */
        rc = SetStartTitle( "Start Group" )

        /* title text for the running list  */
        rc = SetRunningTitle( "Running Group" )

        /* Add an OS/2 program. Arguments are:
        title, startup dir, parameters, exe */
        rc = AddOS2Program( "CMD",,,"CMD.EXE" )

        /* is configured for DOS? */
        if QueryDOSCapable() then do
          /* Add a DOS program. Arguments are:
          title, startup dir, parameters, settings stem */
          drop settings
          settings.0 = "DPMI_DOS_API=ENABLED"
          settings.1 = "DPMI_MEMORY_LIMIT=8"
          rc = AddDOSProgram( "DPMI DOS Session",,, "settings" )

          /* Add another DOS program. Arguments are:
          title, startup dir, parameters, settings stem */
          drop settings
          settings.0 =  'DOS_HIGH=1'
          settings.1 =  'DOS_UMB=1'
          settings.2 =  'VIDEO_MODE_RESTRICTION=CGA           '
          rc = AddDOSProgram( "Big DOS Session","","", "settings" )
```

Figure 216 (Part 1 of 2). Sample REXX Code to Modify PGMSHELL.EXE

```
        /* Add winos2 */
        rc = AddDOSProgram( "WinOS2","","/c winos2", "" )
      end

      /* add shutdown option. arguments: title, completion msg */
      rc = AddShutdown( "Shutdown", "Shutdown Complete" )
    end
    otherwise do
      say 'unexpected execution environment'
      return 3
    end
  end
  return 0
```

Figure 216 (Part 2 of 2). Sample REXX Code to Modify PGMSHELL.EXE

E.2.2 Available REXX Functions

There are several REXX functions that are available under the PGMSHELL environment:

- SetStartTitle(<start group title>) - Sets the title text above the list of startable programs.
- SetRunningTitle(<running group title>) - Sets the title text above the list of running programs.
- QueryDOSCapable() - Returns a Boolean indicating if the system is configured to emulate DOS.
- AddOS2Program(<session title>, <startup directory>, <program param eters>, <program executable>) - Adds an OS/2, protected mode program to the start list.
- AddDOSProgram(<title>, <startup directory>, <command.com arguments >, <DOS settings stem variable>) - Adds a DOS mode program to the start list. ALL elements of the stem variable are considered DOS settings. Therefore, drop the stem variable before assigning DOS settings to it.
- AddShutdown(<shutdown title>, <shutdown complete message>) - Adds the shutdown option to the start group

E.3 MShell

You will find MShell.zip on the diskette that was included with this book. MShell is an alternative, simple, mini shell for OS/2 2.X that uses the replaceable shell architecture of the Workplace Shell. With this architecture and their own EXE, programmers can easily make OS/2 a turnkey platform.

The following steps will get MShell installed:

1. Unpack the MShell.zip file with PKUNZIP.
2. Copy MSHELL.EXE into the root directory of the boot drive.
3. Edit your CONFIG.SYS and modify the RUNWORKPLACE statement to look like the following example:

   SET RUNWORKPLACE=C:\MSHELL.EXE

4. Reboot.

If your boot drive is not C: you will have to modify the RUNWORKPLACE accordingly with the correct path information.

MShell is a program launcher and provides one list of programs that are available for you to click on and start. When MShell.exe runs, it will look for MShell.ini in the root directory of the boot drive. If there is no MShell.ini, in the root directory, MShell will create one for you. You can customize the list, that appears in the MiniShell, by editing MShell.ini and adding or removing any applications you would like. We are including the MShell.ini file we created, by editing the default INI and adding the applications we wanted.

```
* MShell.ini defines programs that MShell can start
* Configure MShell.exe using the RUNWORKPLACE setting in CONFIG.SYS
* MShell.exe finds its INI in the root of the boot drive
* Each line in the INI file has two parts:
*  1: program title text to appear in MShell window
*  2: the CMD.EXE start command to start the program
* Separate the parts with a semicolon
* Lines starting with ! start automatically at bootup
* Comment lines begin with  *

* command processors
OS/2 Command Prompt; start /f
DOS Command Prompt; start /f /fs /dos
Win-OS/2; start /fs /dos /c winos2

* other stuff
Help for Start; start help start
Solitaire; start /f klondike
Norton; e: & cd\ncpm & start ncpm.exe
Corel; e: & cd\corel32 & start coreldrw.exe
WinWord; e: & cd\winword & start winword.exe

* example to start from other than C:\
* PM Program; d: & cd \pmpgm & start pmpgm
```

Figure 217. Example of our MShell.ini

MShell.ini contains program-start information which MShell reads and then it displays the program information in a listbox. Unlike most INI files, MShell.ini is plain text and you can use a normal text editor when changing MShell.ini.

There are two parts to an MShell.ini line:

- the program title to display
- a START command acceptable to OS/2's command processor CMD.EXE.

You have to separate these two parts with a single semi-colon (;). Here is an sample MShell.ini line which will start the Klondike Solitaire program:

```
Solitaire ;  start /f "Solitaire" klondike.exe
```

Lines in the INI that begin with an exclamation mark (!) will be automatically started by MShell when the system boots. The example that follows will start the PMCLOCK when the system comes up:

```
!Clock ;  start pmclock
```

If you want to start programs from a drive or directory other than the root directory, you have to use the command concatenator for CMD.EXE. This is the ampersand (&). An example INI line follows:

```
Save Changed Files;  d: & cd \wp & xcopy *.* a: /m
```

In the example the ″d:″ is the drive you wish to copy from. The ″cd \wp″ is changing to the directory you wish to copy the files from. The ″xcopy *.* a: /m″ will copy any file in the current directory, that has it′s archive bit on, to drive a:. Any comment lines in MShell.ini must begin with an asterisk (*).

MShell can start up to a maximum of 50 programs or batch files. This is an MShell limit, not an OS/2limit.

In addition to starting programs MShell has several options that you can select by clicking on the ″Options″ selection on the MiniShell action Bar. The options you will find there are:

- Print spooler - presents a dialog which allows interaction with the OS/2 spooler, and lets you Pause, resume, and delete jobs or Pause and resume print queues.
- Save Desktop - will broadcast a posted WM_SAVEAPPLICATION to all children of HWND_DESKTOP, the desktop window. Most PM applications will then save their current settings to the INI file.
- Refresh programs in INI - allows you to re-load MShell.ini at any time. This is good for testing your modifications to MShell.ini without having to reboot.
- Command prompt - starts a windowed OS/2 command prompt (CMD.EXE) session.
- Shutdown system - prompts you to confirm the shutdown. If you Respond Yes the active applications will be closed and the n the system will be shutdown and you can either turn it off or reboot it.

One problem that MShell has is Certain DOS settings can only be specified prior to the DOS session starting. Since the Workplace Shell DOS settings dialog is not available under MShell you will not be able to set the DOS settings with out the use of some sort of program.

MShell has limited function so if you have to install an OS/2 PM printer driver, you will have to do it while the normal Workplace Shell is still up, prior to booting with the MShell. Once MShell is installed you can use the spooler to manipulate the print queues.

Appendix F. GG24-4201 Diskette Contents

Following is a directory listing of the files on the diskette. We have noted some exceptions but, in most cases, the ZIP files contain the source and .EXE of the named file. The files were compressed using PKZIP**. Also note that the final diskette was prepared after the book was submitted for printing. Some of the file sizes and dates may not match those on the diskette.

```
CHGDT    ZIP   106279   1-24-94   7:45p
CMDS     ZIP     1752  12-10-93   2:07p
MSHELL   ZIP    57777   4-17-94  10:16p MSHELL.EXE see Appendix E
DELALL   ZIP    18049   1-24-94   7:42p Utility to erase all dirs/files
DTUPD    ZIP    34203   1-24-94   7:46p
INF      ZIP   478918   4-18-94   3:51p INF version of book (use VIEW)
INI2DESK ZIP    18171   1-24-94   7:43p
INITEST  ZIP    56293  12-10-93   2:05p
INI_REXX     <DIR>      4-16-94   8:36p REXX programs from .INI chapter
TSHELL   ZIP    45835   4-17-94  10:16p TSHELL.EXE see Appendix E
SOURCE   ZIP    22815   4-18-94   2:36p MSHELL source see Appendix E
QDESK    ZIP    30763   1-24-94   7:48p DLL for desktops utility
QFOLDER  ZIP      918  12-10-93   2:09p
QOBJ     ZIP    14541   1-24-94   7:49p
RC_FILES     <DIR>      4-16-94   8:37p .RC files described in book
RUN      ZIP   173726   4-16-94   8:15p DT creater/switcher run modules
SHUTDOWN     <DIR>      4-16-94   8:39p TSR Shutdown program and sourc
VPR_DESK     <DIR>      4-16-94   8:39p VisProREXX (c) Appendix A source
        20 file(s)    1060040 bytes used
                        34816 bytes free
```

INF Version

You will find an .INF version of this book on the enclosed diskette. In order to get it to fit on the diskette the bimaps are in black and white. I felt that you could at least benefit from the text and table information using the search and hyper-text facility of the OS/2 VIEW utility while tolerating the ″scratchy″ pictures. To use the .INF, copy INF.ZIP to your system and unpack it with PKUNZIP.EXE. Then type VIEW GG244201 at an OS/2 command line while in the directory of the .INF file.

This .INF file is protected under the copyright of this book. If you have to distribute it to someone on an emergency basis, please ensure that the recipient also places an order for the book. Thank you, JASII.

Index

Special Characters

A

B

C

D

E

F

G

H

I

K

L

M

N

S

T

U

V

W

X