User's Guide

Borland®
Resource Workshop™
for OS/2®

Resource Workshop®
User's Guide

Borland® C++ for OS/2®

Version 1.5

Borland International, Inc.
100 Borland Way, Scotts Valley, CA 95066-3249

PRINTED IN THE UNITED STATES OF AMERICA

1E0R0294
9495969798-987654321
H1

Contents

Tables

Figures

Introduction

Resource Workshop is a sophisticated tool that integrates the entire process of designing and compiling resources for applications running under OS/2, Version 2.0 and later. This manual describes all the Resource Workshop tools and how to use them, and also provides general information on designing and using resources in OS/2 Presentation Manager (PM) applications.

If you write applications that run under OS/2 PM, or if you want to modify the visual interface of PM applications written by others, Resource Workshop is the easiest and most powerful way to get your programs looking the way you want.

Resource Workshop features

Resource Workshop provides everything you need to create and modify OS/2 resources, including graphics-oriented visual resource editors that make it easy to design and modify resources and a script editor for manipulating resource scripts

Among other things, Resource Workshop

- Makes it easy to manage hundreds of resources stored in dozens of files.
- Performs mundane tasks for you, such as automatically loading the correct editor when you choose a resource, inserting references to resource files as necessary in your .RC file, and adding **#define**s or constants for your resource IDs to the appropriate files.
- Includes extensive, multilevel Undo and Redo features that let you step back through changes you've made.
- Includes all the compilers you need and makes it easy to compile your resources only when you need to.
- Decompiles binary resource files, so you can change a program's resources even if you don't have access to the source code.
- Includes features that automatically check for errors, making it easy to test resources for errors like incorrect syntax and duplicate resource IDs.

Resource Workshop is easy to use, so even those with limited programming experience can design user interfaces for application programs.

Hardware and software requirements

The following are the minimum requirements for using Resource Workshop:

- You must have a computer capable of running OS/2 2.0 (80386 processor or higher). The computer must have at least 4MB of RAM and a graphics display and adapter (Hercules, EGA, VGA, or better). A mouse or other pointing device is also required.
- OS/2 2.0 or later must be installed on your computer.
- You must have at least 3.5MB of free disk space.

What's in the manual

This manual explains how to use Resource Workshop to develop PM resources. It doesn't tell you how to write PM programs or how to write code in your programs to access resources. The manual assumes that you know the basics of PM programming.

The first part of this manual explains what PM resources are and how Resource Workshop manages them in projects.

- Chapter 1, "Getting started," describes how to install, start, and exit Resource Workshop and how to access the Help system.
- Chapter 2, "Resource Workshop basics," gives a brief introduction to the different kinds of resources available under OS/2 PM, the kinds of editors used in Resource Workshop to edit them, and the different kinds of files you can store the resources in. It also introduces the notion of a project, which includes all the resources for a given program.
- Chapter 3, "Working with projects, resources, and identifiers," covers projects in more detail, describing how to set up and use projects, edit and add resources, and coordinate the identifiers used in your resources with those in your program.

The remaining chapters describe the different resource editors contained in Resource Workshop.

- Chapter 4, "Using the Dialog editor," covers the Dialog editor, including all aspects of creating and modifying dialog boxes and the controls they contain. The Dialog editor lets you design, modify, and test your dialog boxes outside your program.

- Chapter 5, "Using the Bitmap editor," provides the basics for working with the Bitmap editor. Resource Workshop starts the Bitmap editor when you choose a bitmapped resource—an icon, a cursor, or a bitmap.
- Chapter 6, "Using the Script editor," explains how to use any other kinds of resources you might want to define. All the resources described in earlier chapters are the standard ones defined and handled by OS/2. If these resources don't meet your needs, you can use the resource mechanism to create user-defined resources that store other kinds of resource data for your programs.
- Appendix A, "Technical notes," provides technical notes on a number of aspects of Resource Workshop, including compatibility with the IBM Resource Compiler and use of dialog boxes as child windows.
- Appendix B, "Borland PM Custom Controls," describes the BPMCC library and how to use to the library to create customized dialog controls such as buttons, check boxes, shading, and the like.

Online documentation

The Resource Workshop disk set includes two files that contain information that was not available when the manual went to press or that goes beyond the scope of the manual. These files are copied onto your hard disk by the installation program.

Online text file	Description
BPMCCSTL.RW	This file presents style condiderations you can follow when designing Borland Presentation Manager Custom Control (BPMCC) dialog boxes for your OS/2 based software.
BPMCCAPI.RW	This file describes technical aspects of the Borland Presentation Manager Custom Controls (BPMCC) and contains information that might be useful or of interest to the advanced resource designer. However, you can successfully create or modify application resources for BPMCC using the information contained in the file MANUAL.RW.
MANUAL.RW	This file, which is included only if needed, contains additions and corrections to the manual. If MANUAL.RW is present, you should be familiar with its contents before you use Resource Workshop.

Conventions, symbols, and special fonts

This manual uses the following conventions, symbols, and special fonts:

`Monospaced type` This typeface represents text as it appears onscreen or in a program. It is also used for anything you must type literally (such as `DIALOG` in a resource script).

ALL CAPS — All capital letters are used for the names of files and C++ constants.

[ ] — Square brackets [ ] in text, syntax statements, or OS/2 command lines enclose optional items. *Text of this sort should not be typed verbatim.*

Boldface — Boldface type indicates

- Function names (such as **printf**), class, and structure names when they appear in text (but not in program examples).
- Reserved words (such as **char** and **switch**)
- Command-line options (such as **–30**).

Italics — *Italic* type indicates variable names in C++, method types, or method names that appear in text, or identifiers in syntax statements. This typeface is also used to emphasize certain words, such as new terms.

Keycaps — This typeface indicates a key on your keyboard. For example, "Press *Esc* to exit a menu."

Key1+Key2 — Key combinations produced by holding down one or more keys simultaneously are represented as *Key1+Key2*. For example, you can paste something into one of the editors by holding down the *Shift* key and pressing the *Ins* key. This key combination is represented as *Shift+Ins*.

Command1|Command2 — This command sequence represents a choice from the menu bar followed by a choice from the menu displayed by the menu bar command. For example, "Choose File | Open" means "Display the File menu and then choose the Open command."

 This arrow icon indicates material you should take special notice of.

Contacting Borland

The Borland Assist program offers a range of services to fit the different needs of individuals, consultants, large corporations, and developers. To receive help with your questions about our products, send in the registration card. North American customers can register by phone 24 hours a day by calling 1-800-845-0147.

Borland Assist plans

Borland Assist is made up of three levels of support:

- Standard Assist gives all registered users assistance with installation and configuration, and offers automated and online services to answer other product questions (see the following table).
- Enhanced Assist plans are designed for individuals who need unlimited support on a toll-free number or priority hotline access.

- Premium Assist plans are designed to support large corporations and software developers.

Available at no charge, Standard Assist offers all registered users the following services:

Service	How to contact	Cost	Available	Description
Installation hotline	408-461-9133	The cost of the phone call	6:00am – 5:00pm PST Monday – Friday	Provides assistance on product installation and configuration.
Automated support	Voice: 1-800-524-8420 Modem: 408-431-5250	Free The cost of the phone call	24 hours daily	Provides answers to common questions. Requires a Touch-Tone phone or modem.
TechFax	1-800-822-4269 (voice)	Free	24 hours daily	Sends technical information to your fax machine (up to 3 documents per call). Requires a Touch-Tone phone. Document #1 is the catalog of available catalogs.
Online services				
Borland Download BBS	408-431-5096	The cost of the phone call	24 hours daily	Sends sample files, applications, and technical information via your modem. Requires a modem (up to 9600 baud).
CompuServe	Type `GO BORLAND`. Address messages to Sysop or All.	Your online charges	24 hours daily; 1-working-day response time	Sends answers to technical questions via your modem. Messages are public.
BIX	Type `JOIN BORLAND`. Address messages to Sysop or All.	Your online charges	24 hours daily; 1-working-day response time	Sends answers to technical questions via your modem. Messages are public.
GEnie	Type `BORLAND`. Address messages to All.	Your online charges	24 hours daily; 1-working-day response time	Sends answers to technical questions via your modem. Messages are public.

For additional details on these and other Borland services, see the *Borland Assist Support and Services Guide* included with your product.

C H A P T E R 1

Getting started

This chapter describes how to install Resource Workshop and covers the basics for starting and exiting the program and for getting Help.

Installing Resource Workshop

Resource Workshop is automatically installed as part of the Borland C++ for OS/2 installation process. If you later need to reinstall Resource Workshop, run the INSTALL program on your Borland C++ for OS/2 installation disk. For more information about installation, see the README file on the same disk.

Starting Resource Workshop

Once you've installed Resource Workshop and it appears as an icon on the desktop or in an OS/2 icon view, double-click the icon to start Resource Workshop and display the Resource Workshop Project window.

You can also start Resource Workshop in the following ways:

- Using the drag-and-drop technique, select a resource or project file icon and use the right mouse button to drag it to the Resource Workshop icon. When a square outline appears around the Resource Workshop icon, release the right button to drop the resource or project into Resource Workshop.
- From the OS/2 File Manager, you can double-click the icon for a project file or resource file. The project or resource file type must be associated with the Resource Workshop application.
- From the command line in an OS/2 window, enter `WORKSHOP`, optionally followed by the name of the file you want to load. The next section describes the available command-line options.

Using command-line options

When you start Resource Workshop from the OS/2 command line, there are a number of options you can use. The command-line format is as follows:

```
WORKSHOP [option [optionarg] ... option [optionarg]]
```

- An *option* is one of the command-line switches listed in Table 1.1. Options must be preceded by a dash (-) or a slash (/).
- An *optionarg* is the argument to an option, such as the path name that follows the **–i** option.
- The command-line text can be entered in uppercase or lowercase letters or any combination of the two (see the examples in Table 1.1).

Table 1.1 Resource Workshop command-line options

Option	Description
–x	Clears the include path. If you've set an include path using the Preferences dialog box (see page 17), this option erases your settings.
–i *pathname*	Adds an additional path specification to the include path. For example: `WORKSHOP -i c:\mystuff\include`
–fo *filename*	Sets the .RES Multi-Save option to the specified file name (see the Preferences dialog box on page 17). With this option set, whenever Resource Workshop compiles the current project, it also saves the resources in binary format to the indicated .RES file. For example: `WORKSHOP -FO C:\MYSTUFF\MYPROJ.RES C:\MYSTUFF\MYPROJ.RC`
–fx *filename*	Sets the Executable Multi-Save option to the specified file name (see the Preferences dialog box on page 17). With this option set, whenever Resource Workshop compiles the current project, it also binds the resources in binary format to the indicated .EXE file. For example: `workshop /fx c:\mystuff\myproj.exe c:\mystuff\myproj.rc`

The following command line starts Resource Workshop and clears the include path:

```
C:\WORKSHOP\WORKSHOP -X
```

Getting Help

To get Help in Resource Workshop, you can do any of the following:

- To bring up the Help menu, press *Alt+H* or select the Help command on the main menu.
- To display the Help index directly, without going through the Help menu, press *F1*.
- To get context-sensitive Help, press *F1* after you select an icon or window or after you hold down a mouse button over a menu command.

Help runs as a separate application under OS/2. You can leave Help running and return to Resource Workshop, or you can terminate Help altogether by pressing *Alt+F4*, double-clicking the Control-menu box, or choosing File | Exit.

In addition to Help, you can use the status line at the bottom of the Resource Workshop window for explanations of Resource Workshop functions.

When you're in the menus in the Project window or a resource editor, the status line provides a brief explanation of the highlighted menu command. Other status line displays are described in the chapters on the specific resource editors.

Exiting from Resource Workshop

To exit from Resource Workshop, switch to the Project window and either choose File | Exit or double-click the Control-menu icon (the top left corner of the Project window).

If you've made any changes you haven't saved, Resource Workshop asks if you want to save the changes before quitting.

C H A P T E R 2

Resource Workshop basics

This chapter provides an overview of Resource Workshop and OS/2 Presentation Manager (PM) resources, including the following topics:

- Understanding Presentation Manager resources
- The types of resource files
- Configuring Resource Workshop

Understanding Presentation Manager resources

This section answers some basic questions about PM resources, specifically

- What are resources?
- What resources can I edit?

The second section includes descriptions of all the PM resource types.

What are resources?

Resources are the visible portions of your PM program—the dialog boxes, menus, bitmaps, string tables, pointers, and so on. For example, when you open a dialog box and click a button, you're interacting with the application's resources. In PM applications, resources provide a consistent user interface that makes it easy for users to switch from one PM program to another.

The following is an example of a resource—a dialog box that lets users make certain choices:

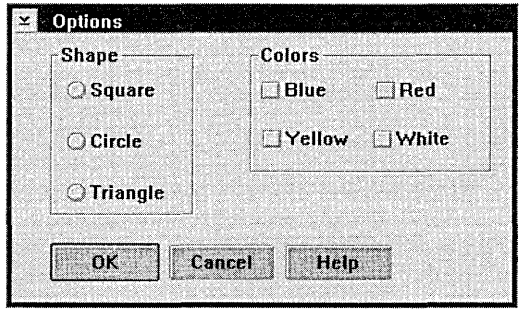


Figure 2.1
Typical dialog box

The entire dialog box and all the controls in it are defined in the PM program as resources.

Resources in a PM application can be created separately from the program code, letting you make significant changes to the user interface without opening the file that contains the program code.

For example, if you're writing a PM financial application, you would keep the code for your financial algorithms in one set of files and your resources in another. This allows you to compile the program code independently of the resources, and you (or whoever is responsible for user interfaces) can modify the resources without affecting the code that handles the financial calculations. When you're ready to build your program, you use Resource Workshop to bind the resources to the executable file.

Also, if your applications use the same set of resources, you can use the dialog boxes, icons, and customized pointers in your resource files over and over, or you can use your existing resource files as the starting point for new resource files.

What resources can I edit?

Resource Workshop supports the following PM resources:

- Accelerator tables
- Bitmaps
- Dialog boxes
- File-association tables
- Help tables
- Help subtables
- Icons
- Menus
- Message tables
- Pointers
- String tables
- User-defined and RCDATA resources

Accelerator tables

Chapter 6 describes how to create accelerator tables.

Accelerators (sometimes called shortcuts or hot keys) are keyboard combinations a user typically presses to execute a menu command, instead of displaying the menu and clicking the command name. For example, a PM program can use the accelerator *Shift+Ins* for the Paste command, which pastes text or images from the Clipboard. Accelerators typically appear in the menu to the right of the commands to which they're linked. You can also create accelerator resources that define new functions not available from your program's menus.

Using Resource Workshop, you write a resource script to create an *accelerator table*. The resource type name is ACCELTABLE.

Bitmaps

Chapter 5 describes how to create or change bitmaps.

A *bitmap* is a binary representation of a graphic image in a program. Presentation Manager itself uses bitmaps for many of the images representing controls on a typical window—for example, scroll bar arrows, the title bar icon, and the Minimize or Hide button.

A single bitmap resource can contain multiple *images* targeted to specific screen resolutions and color formats.

The following figure shows an example of a bitmap, the Paint Can tool from the Bitmap editor Tools palette:

Figure 2.2
Paint Can tool bitmap

You create bitmaps using Resource Workshop's Bitmap editor. You also use this editor to work with the other specialized bitmapped resources: icons and pointers.

Dialog boxes

Chapter 4 describes how to create dialog boxes.

A *dialog box* is a window (usually a top-level window) that communicates information to the user and lets the user select choices, such as files to open, colors to display, text to search for, and so on. The resource type name is DLGTEMPLATE.

A dialog box like the one in Figure 2.1 usually includes *controls*, such as radio buttons, check boxes, and push buttons. Resource Workshop makes it easy to put any combination of controls in a dialog box and lets you test your dialog box and debug its behavior before binding it to your executable code.

File-association tables

Chapter 6 describes how to create file-association tables.

A *file-association table* links data files to the applications that can edit them. When the user selects a data file from the File Manager, PM automatically starts the associated application and loads the selected file.

Using Resource Workshop, you write a resource script to create a file-association table. The resource type name is ASSOCTABLE.

Help tables and subtables

Chapter 6 describes how to create help tables and subtables.

A *help table* contains entries that let the application gain access to requested help data for application windows, dialog boxes, and message boxes. The entries in the help table point to further entries in a help subtable, which in turn point to the actual help text.

A *help subtable* contains entries for each item in the application window (control, child window, or menu item) for which help is available.

Using Resource Workshop, you write resource scripts to create help tables and subtables. The resource type name is HELPTABLE or HELPSUBTABLE.

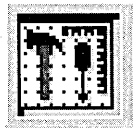
Icons

Chapter 5 describes how to create or change icons.

Icons are small bitmaps—40×40 (supported on very high-resolution devices), 32×32, or 16×32 pixels in size—that typically represent minimized windows. You create icons using the Bitmap editor.

To see what icons look like, look at the OS/2 System, Master Help Index, and Resource Workshop icons on the OS/2 desktop. When you minimize a window, it usually changes to its icon form—either at the bottom of the desktop or inside the Minimized Window Viewer window—depending on its current window settings.

Figure 2.3
The Resource Workshop icon



A single icon resource can contain multiple *images* targeted to specific screen resolutions and color formats.

In OS/2 Presentation Manager 2.0, unlike previous versions, icons and pointers (described shortly) are functionally identical.

The *default icon* has an ID of 1, and gets put in the extended attributes for the file. The default icon is the one that shows up as the program icon in the Workplace Shell.

Menus

Chapter 6 describes how to create a menu script.

PM programs usually include a menu bar that lists the names of individual menus. A typical menu contains one or more menu items (commands). For example, most PM programs have a File menu with commands for creating, opening, saving, or printing files.

Using Resource Workshop, you create menus by writing a simple text script.

Message tables

Chapter 6 describes how to create message tables.

A *message table* provides one means of storing string data—like status lines and error messages—in your application. (You can also store string data in a string table.)

Using Resource Workshop, you create message tables by writing a text script. The resource type name is MESSAGETABLE.

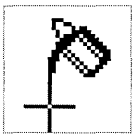
Pointers

Chapter 5 describes how to create or change pointers.

A *pointer* is a bitmapped image that represents the position of the mouse on the screen. PM programs use customized pointers to indicate what type of task the user is currently performing. You create pointers using the Bitmap editor.

You can see an example of customized pointers in the Resource Workshop Bitmap editor. Each time you choose a new painting tool and move the pointer to the image you're working on, the pointer takes the shape of the selected tool. For example, if you click the Paint Can in the Bitmap editor Tools palette and move the pointer to the image, the pointer becomes a paint can.

Figure 2.4
Paint can pointer in Bitmap editor



A single pointer resource can contain multiple *images* targeted to specific screen resolutions and color formats.

In OS/2 Presentation Manager 2.0, unlike previous versions, pointers and icons (described previously) are functionally identical.

String tables

Chapter 6 describes how to create string tables.

String tables contain text (like descriptions, prompts, and error messages) that's displayed as part of a PM program. Because these text strings are PM resources that are separate from the program (instead of strings embedded in the program), you or others can edit and translate messages displayed by a program without having to make any changes to the program's source code.

User-defined and RCDATA resources

Chapter 6 describes how to create user-defined and RCDATA resources.

User-defined resources and *RCDATA* resources (essentially the same in Resource Workshop) consist of any data you want to add to your executable file. For example, if you have a large block of initialized, read-only data, such as a text file, you can add it to your executable file as a user-defined resource.

Types of resource files

A file you create and edit with Resource Workshop can be in either binary or text format. In addition, Resource Workshop can generate standard OS/2 file formats, which means you can use Resource Workshop files with

programs that generate binary code from resource script files, like BRCC and the IBM Resource Compiler.

Some Resource Workshop .RC files can be incompatible with the IBM Resource Compiler. The incompatibilities between Resource Workshop and the IBM Resource Compiler are described in Appendix A (pages 107-110). Instead of using the IBM Resource Compiler, you can compile to .RES format with Resource Workshop or use the BRCC command-line resource compiler.

Note that, although Resource Workshop can read files in the OS/2 Presentation Manager Version 1.2 format, it saves files only in the Version 2.0 format.

Resource compiler files

A resource compiler (.RC) file is a text file containing definitions of one or more resources. The file can contain resources defined in script form and references to other files containing resources.

See Help for information on the script commands.

In general, you should base all your Resource Workshop projects on at least one .RC file.

Compiled resource files

A compiled resource (.RES) file contains one or more compiled resources.

Typically, when creating a PM program, you compile all resources for an application into a single .RES file, and then bind the .RES file to the executable file as part of the linking process. However, you don't have to produce a .RES file, because Resource Workshop can compile resource files and bind them directly to an executable file.

Executable files

An executable (.EXE) or dynamic-link library (.DLL) file is the ultimate destination for all resources you define with Resource Workshop. Usually, you compile an .RC file into a .RES file, then use your compiler to bind the .RES file to the executable or .DLL file. You can also use Resource Workshop to bind the resources directly to the executable or .DLL file and bypass the resource compiler altogether.

You can't create a new executable or DLL file with Resource Workshop.

If you want to change the resources in a compiled binary file (an executable file, a DLL file, or a .RES file), Resource Workshop will decompile the file and let you make changes, and then save the resources back to the original binary file.

Dialog files

A dialog (.DLG) file is a resource script (text) file that typically contains descriptions of one or more dialog boxes. There is, however, no requirement that a .DLG file contain dialog boxes; it can contain any of the resources found in an .RC file.

Bitmapped resource files

Resource Workshop supports three kinds of bitmapped resource files:

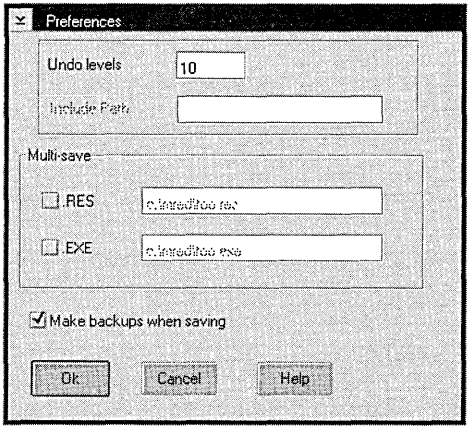
- A bitmap (.BMP) file contains a bitmap resource in binary format.
- An icon (.ICO) file contains an icon in binary format.
- A pointer (.PTR) file contains a customized pointer in binary format.

Icons and pointers in OS/2 Presentation Manager 2.0 are functionally identical.

Configuration preferences

To configure Resource Workshop to your preferences, choose File | Preferences. Resource Workshop displays the Preferences dialog box.

Figure 2.5
Preferences dialog box



The choices you can make in this dialog box are described in the following sections.

Undo Levels

Resource Workshop has a multilevel Undo and Redo feature that lets you correct actions in any of the resource editors. Depending on the amount of available memory in your computer, you can undo or redo up to 99 actions. The default number of levels is 10.

For example, if you're working in the Bitmap editor and you fill an area with a color, then select and move a portion of the image, only to find that the fill doesn't look right, you can undo the move and then undo the fill. If you want to see how the fill looks again, you can redo the fill and then redo the move. (For each undo, press *Alt+BkSp* or choose Edit | Undo; for each redo, press *Shift+Alt+BkSp* or choose Edit | Redo.) You can work your way

back and forth through your edit session this way through as many levels as you have set in the Undo Levels entry field.

The number of undo levels can be limited by available system memory. A memory-intensive resource—like a large bitmap with several flood fills—can cause fewer undo levels to be available.

Include Path

This is the path Resource Workshop will search for files containing identifiers or resources. You can set this option only if you do not have a project open. When you choose this option, Resource Workshop saves it in WORKSHOP.INI as the default include path.

Multi-Save

The .RES and .EXE preferences control how a project is to be saved when you select File | Save Project. These preferences are enabled only when a resource compiler (.RC or .DLG) project is open because they apply only to a specific project. Regardless of the Multi-Save settings, the project always gets saved in its original format as well. (For example, if the project is an .RC file, the resources in the file are always saved as resource scripts in addition to any Multi-Save options.)

If you choose .RES, Resource Workshop compiles the project's resources and saves them in .RES format (in binary format).

If you choose .EXE, Resource Workshop compiles the project's resources and binds them to the executable file specified in this option (can be an .EXE or .DLL file).

Make Backups When Saving

If you check the Backups option, Resource Workshop creates an additional set of backup files each time you save a project. Backup files have a tilde (~) as the first character in the file extension. For example, if you save a file called NEWTOOL.BMP, the backup file is called NEWTOOL.~BM.

C H A P T E R 3

Working with projects, resources, and identifiers

Resource Workshop organizes resources into *projects*. In general terms, a project consists of at least one of the following types of resource files:

- A resource script (.RC or .DLG) file
- A binary resource (.RES) file
- A binary pointer (.PTR or .CUR) file
- A binary icon (.ICO) file
- A binary bitmap (.BMP) file
- An executable (.EXE) file
- A dynamic-link library (DLL) file

For example, if you wanted to create a Resource Workshop project consisting only of a pointer, you could create a Resource Workshop project that contains a single .PTR file.

However, to take full advantage of Resource Workshop's project management capability, you should build your projects around an .RC file. For example, a single .RC file can contain or refer to several dialog boxes, pointers, and icons, as well as scripts for menus and accelerator tables.

This chapter covers the three main aspects of Resource Workshop projects:

- Managing a project
- Working with resources
- Working with identifiers

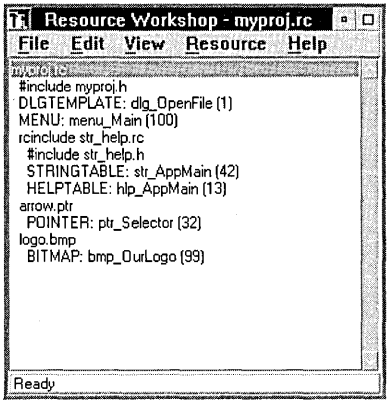
Managing a project—the Project window

When you open a project in Resource Workshop, the central focus is the Project window. You can open many instances of the editors—Dialog editor, Bitmap editor, and Script editor—within a single project, and the single Project window serves as the axis around which all your work revolves.

Many projects consist of a main .RC file that contains references to other resources. The references in the main .RC file can include

- Other .RC files
- .DLG files
- Binary resource files, such as .BMP (bitmap), .ICO (icon), and .PTR (pointer) files
- .H (header) files containing **#define**s that assign meaningful names to your resources

Figure 3.1
Typical Project window



This view of MYPROJ.RC is displayed when you choose View | By File. It contains the following:

- A **#include** for the header file in which the project's identifiers are stored
- Two embedded resources, of type DLGTEMPLATE and MENU
- An **rcinclude** reference to STR_HELP.RC, which contains a reference to STR_HELP.H, as well as data for a string table and a message table
- Two linked bitmapped resource files:
  - A pointer file called ARROW.PTR, which contains data for a pointer
  - A bitmap file called LOGO.BMP, which contains data for a bitmap

In addition to providing you with an overview of all of the files and resources contained in the project, Resource Workshop's Project window also updates and recompiles resources as required. For example, if you change a resource identifier, Resource Workshop automatically recompiles all resources affected by the change.

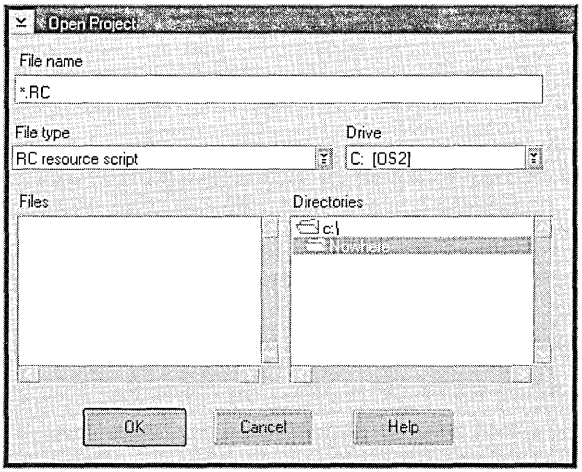
Opening an existing project

An existing project can be one that you created with Resource Workshop or an .RC file you created with other resource development software. If you have access only to an executable file, Resource Workshop can decompile the resources bound to that file so that you can make changes to them.

To open an existing project, do the following:

1. Choose File | Open Project. Resource Workshop displays the Open Project dialog box.

Figure 3.2
Open Project dialog box



To see all the types of files you can open, click the File Type list box down-arrow button. In most cases you'll probably work with .RC files, but you can open any of the listed file types.

If you type one of the standard extensions in the File Name field (instead of picking a file type from the list and letting Resource Workshop insert the extension for you), Resource Workshop assigns the proper file type to the file. However, if you use a nonstandard extension (such as .MNU for a file that contains a menu), be sure to choose the correct file type from the File Type list before loading the file. (In the case of the .MNU file, if the resource is a menu stored as a resource script, the file type would be RC Resource Script.)

2. Specify the file containing the project you want to open by doing either of the following:

   - Type the file name and press *Enter*. If the file isn't in the current directory, you must also specify a path. For example, you might type `C:\RW\MYPROJ.RC`.

- Choose a file from the Files list. For example, if you want to open C:\RW\MYPROJ.RC, select *.RC under File Type, select the appropriate folder icons under Directories until the RW directory is displayed, and then double-click MYPROJ.RC in the Files list.

What Resource Workshop does next depends on whether the project is a binary file or a file containing resource data.

- If the project is a binary file (an executable file, a .RES file, or a DLL file), Resource Workshop decompiles the resources and shows you its progress on the left side of the status line at the bottom of the display.
- If the project consists of an .RC file and other files containing resource data (as is usually the case) or a single resource file containing resource data, Resource Workshop reads the project file to determine all the files in the project. Resource Workshop then works its way through all the files, following references to any additional files, and compiles each resource, showing you its progress in the Compile Status dialog box.

  You can click the Cancel button to cancel the compilation.

  If the compiler encounters an error, Resource Workshop displays the Compiler Error dialog box, which shows you the error and highlights the line where the error occurred.

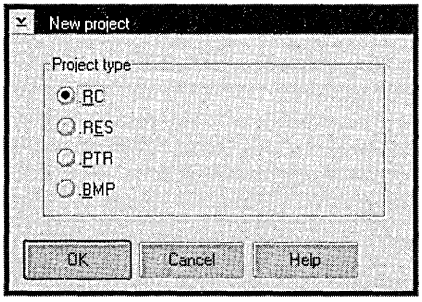
Once the project is compiled or decompiled, Resource Workshop displays the Project window with all the resources listed in it.

Creating a new project

To create a new project, do the following:

1. Choose File | New Project. Resource Workshop displays the New Project dialog box:

Figure 3.3
New Project dialog box



   Decide what type of file you want to base your project on. A typical project is based on an .RC file, because this type of file lets you work with all the available resource types. However, you can also choose one of the following:

- .RES, to work with a binary resource file
- .PTR, to create a project containing a pointer or icon
- .BMP, to create a project containing a bitmap

2. Click the project file type you want, then click OK.

   (If you have a project open, Resource Workshop closes it first. If there are unsaved changes, before closing the project, Resource Workshop asks if you want to save those changes.)

   Resource Workshop displays your new, untitled project in the Project window. You can name the project by using either of the File | Save commands.

Using the Project window

The Project window shows the resources in your project. This section describes several ways you can customize the display, including the following:

- Sorting the Project window
- Previewing resources
- Filtering information

A new project contains no resources yet, so the Project window either lists the available resource types or the project's file name. If you haven't named the project yet, the file name is given as UNTITLED.RC.

When you open an existing project, the Project window lists the resources according to their type or according to their relationship (embedded or linked) to the project file. In this view, the Project window also lists any identifier files or other files included in the project.

The Project window acts as a file management tool, providing an overall view of your project. By scrolling through the Project window, you can quickly scan the resources in the project.

Embedded and linked resources

The resources in your project file can be *embedded* in the file or *linked* to it.

An embedded resource is stored in resource script form in the project file. It exists only as part of the project in which it's stored, and it can't be used in other projects. (You can, however, save an embedded resource to an external file and then link it to another project.)

A linked resource is a separate file that is referenced in the project file. Linked resources can be used in other projects. They can include

- Other projects (.RC or .DLG)
- Binary-format files for bitmaps, icons, and pointers

Sorting the Project window

The project window can show your resources sorted in two ways: by type (the default) and by file. To choose a sorting order, choose either the By Type or the By File command from the Project window's View menu. The commands are mutually exclusive. Choosing one turns off the other.

Viewing *By Type* lists the contents of your project file by resource type. Under each type, the names of any resources of that type are listed. The By Type display doesn't tell you whether a resource is linked or embedded.

Viewing *By File* lists the contents of your project according to the files in which they're located and in the order in which they appear in the source files.

Previewing resources

In a large project with many resources, it can be difficult to remember what all the resources look like. Resource Workshop gives you an easy way to visually browse through bitmapped and dialog resources. If you check the View | Show Preview menu item, the project window divides into two panes. The left pane shows the list of resources in the project and the right pane shows a preview of the currently selected resource.

Filtering information

You can control what kinds of information the Project window includes by checking or unchecking several items on the View menu. Table 3.1 summarizes the items and what they control.

Table 3.1
Project window filter commands

View Menu Command	Description
Show Identifiers	Displays the identifiers (#**defines**) in the project.
Show Resources	Lists the types and names of resources, like BITMAP: airplane. In most cases you'll leave this option checked. Uncheck Show Resources if you want to see file names only, without a list of resources contained in those files or if you only want to look at identifiers.
Show Items	Displays items within individual resources (for example, SUBMENUs and MENUITEMs defined in a menu resource). This view shows an "economy version" of the resource script—without control IDs or menu IDs, control class names, coordinates, or size values.
Show Unused Types	Lists all the resource types, whether or not they are used in the project.

Selecting a resource

To select a resource, use the mouse or the arrow keys to highlight it in the Project window.

- If you've chosen View | By Type, look for the resource type first. The resource is listed by name under the resource type.

- If you've chosen View | By File, look for the resource under its file name, if you know it. The resource name is preceded by the resource type and a colon.

Saving projects, resources, and files

It's a good idea to save your work often. Resource Workshop provides you with a variety of save commands so you can choose exactly what you want to save and how to save it.

File|Save Project

Choose File | Save Project to save everything in your current project. If you're saving a new project that hasn't been named yet, Resource Workshop displays the Save File As dialog box so you can specify a name and directory.

Resource Workshop always saves the project file and any files it references. If you have selected .RES or .EXE from the File | Preferences dialog box (described on page 17), Resource Workshop also compiles and saves to a .RES file or binds the resources to an executable file or DLL.

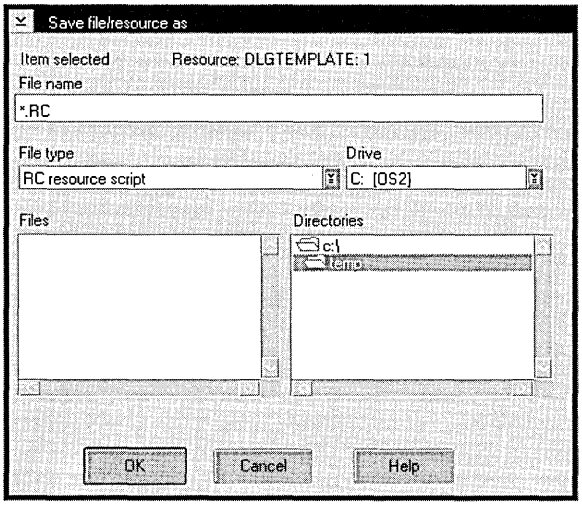
If your project is based on an .RC file, Resource Workshop also compiles the project as part of the save process. This compiled version is stored in a file with an .RWP extension. The next time you open this project, Resource Workshop can save time by loading the already compiled .RWP file instead of having to load and compile the .RC file.

Resource Workshop can do any compiling you might require for your resources. See Figure 3.9 on page 31 for an example of how a project might be compiled.

File|Save File As

If you want to rename the current project or resource file, choose File | Save File As. Resource Workshop displays the Save File As dialog box.

Figure 3.4
Save File As dialog box



Either enter a new file name or choose a file name from the Files list. (In the latter case, Resource Workshop asks if you want to overwrite the existing file.) If you want to put the file in another directory, you can either change the path by using the Directories list or include the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or click OK to save the file.

File|Save All

Choose File | Save All to save everything associated with your project when time is of the essence. Use this command when you are concerned about possible loss of data if you don't move fast—imminent power failure, for example.

When you choose File | Save All, Resource Workshop doesn't query you about possible file overwrites and the like; it saves everything *fast*.

File|Close All

Choose File | Close All to save everything in the current project and then close the project. When it encounters edits to the project that haven't been saved, Resource Workshop queries you about saving them. When everything has been saved, the project is closed and you are presented with an empty Project window.

Resource|Save Resource As

To put a resource in a separate file for use with other projects,

1. Choose Resource | Save Resource As. Resource Workshop displays the Save Resource As dialog box.

This dialog box contains almost the same information as the File Save As dialog box. The primary difference is that, on the third line, instead of showing you the *file* it's about to save, it shows you the *resource* you'll be saving.

2. Either enter a new file name or choose the correct file name from the Files list. If you want to put the file in another directory, you can either change the path by using the Directories list or include the path when you type the file name. When you're satisfied that the file name is correct, press *Enter* or click OK to save the file.
3. Resource Workshop asks if you want the reference in the project file to refer to this external file from now on.
   - Clicking Yes causes the resource script or current file reference in the Project file to be replaced by a reference to the new file. All future changes to the resource will be saved to the external resource file and not in the project file or in any previous resource file.
   - Clicking No creates the resource file without making any changes to the Project file.

Copying resources between projects

There are two ways you can copy a resource from the current project to another project:

- One way is to save the resource as a file, close the current project, open the other project, and add the resource as a file to the new project. If there are any identifiers in the resource, you have to be careful that they're preserved when you add the resource to the new project.
- An easier way is to have two copies of Resource Workshop open, one for each project, and to use the OS/2 Clipboard to copy the resource from one project and paste it to the next. This method is not only faster than the first method, but it also saves all the identifiers.

To copy a resource using the second method,

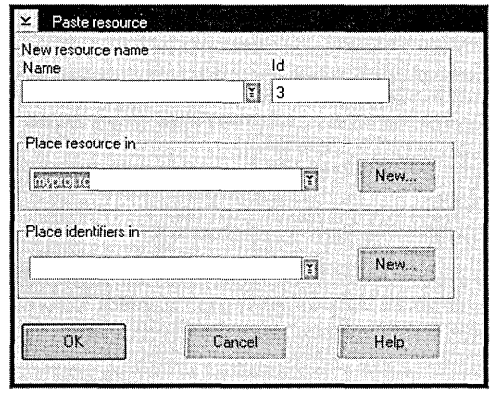
1. Open two copies of Resource Workshop, one with the project containing the resource you want to copy (the source project) and another with the project you want to copy the resource to (the target project).
2. Be sure the target project has a reference to an identifier file that will receive any identifiers in the new resource. (If necessary, choose File | Add to Project and add the appropriate type of identifier file.)

Don't use the *Ctrl+Ins* accelerator—it won't copy the resource properly.

3. Select the source project, then select the resource you want to copy in the Project window. Choose Edit | Copy to copy it to the OS/2 Clipboard.

4. Select the target project and then, with the Project window active, choose Edit | Paste to paste the resource into the project. Resource Workshop displays the Paste Resource dialog box.

Figure 3.5
Paste Resource dialog box



5. The Place Resource In list box should contain the name of the target project. Make sure the Place Identifiers In list box contains the name of the identifier file that will receive any identifiers in the resource (if necessary, scroll the drop-down list and choose the correct file name), then press *Enter* or click OK to paste in the new resource.

Working with binary files

Resource Workshop allows you to open executable files, .RES files, and DLLs as projects so you can customize their user interfaces. For example, you might want to translate your application interface into another language.

When you load one of these files, Resource Workshop decompiles the resources in the file and shows them to you as though they were part of a regular .RC file. When you're finished with your changes, Resource Workshop compiles the resources again into binary code and stores them in the original file.

All resource IDs in binary files must be integers.

Because the resources in a decompiled binary file aren't stored in resource script form, you can't assign identifiers to the resource IDs.

You can, however, save the project as an .RC file. The resources can then be saved as resource scripts, and you can assign identifiers to them.

If you're customizing the user interface of a program and have access only to the executable file or DLL, you might also want to save your changes in a separate .RC file so you can apply the changes to the next version of the program. The resources in your .RC file must have the same resource IDs as their counterparts in the new version and must otherwise be compatible with the new version.

When you save the project as an .RC file, Resource Workshop doesn't automatically save the resources back to the original file unless you've entered the original file name as a Multi-Save option in the Preferences dialog box.

To save a binary file as a project and add identifiers, do the following:

1. Choose File | Open Project and select the executable, .RES, or DLL file from the Open Project dialog box.
2. Choose File | Save File As. In the Save File As dialog box, select RC Resource Script from the File Type list box. Enter the name of the new .RC file.

   When you press *Enter* or click OK to save the file, Resource Workshop automatically places you in the .RC file.
3. Choose File | Preferences and enter the name of the original binary file as a Multi-Save option.
   - If the original binary file was a .RES file, check .RES and enter the name in that field.
   - If the original binary file was an executable or DLL file, check Executable and enter the name in that field.
4. Choose File | Add to Project and specify an identifier file to hold the new identifiers. If the file you specify doesn't exist, Resource Workshop creates a new one for you.

Be sure to preserve the current integer values of the resource IDs.

5. Make your changes to the resources and specify identifiers where you want them. For each new identifier, Resource Workshop asks if you want to save it in the identifier file.
6. When you quit and save the file, Resource Workshop saves both the .RC file and the binary file. If the binary file is an executable file or a DLL, the changed resources are bound into it and are available immediately when you run that program.

A typical project

This section describes a simple, but typical, project.

The starting point is an .RC file you create called MYPROJ.RC. This file will be the central file in your project. You can add as many different types of resources as you want, but everything in your project will be referenced in your .RC file.

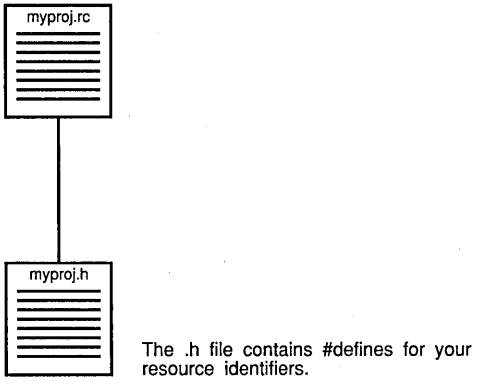
Figure 3.6 MYPROJ.RC, the central project file

The .RC file is the central file in your Resource Workshop project.

See page 38 for a discussion of identifiers.

You can use identifiers for all of your resources. When you add a header (.H) file to the project, Resource Workshop puts a reference to the header file in the .RC file and then knows to store any new identifiers in the header file.

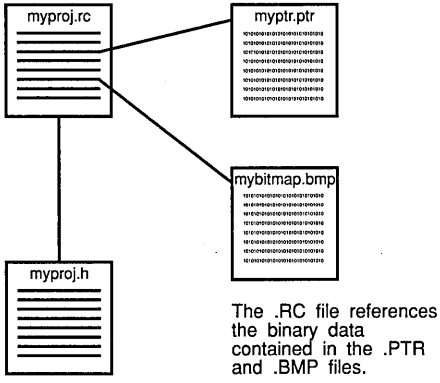
Figure 3.7
MYPROJ.RC points to .H file



Next, you create a pointer and a bitmap and store them in external files, the pointer in a .PTR file and the bitmap in a .BMP file. Resource Workshop puts references to the files in your .RC file. Because they're saved outside the project, you can use these resources in other projects. If you edit the .PTR or .BMP file with the Resource Workshop Paint editor, the changes appear in all projects in which the resource is used.

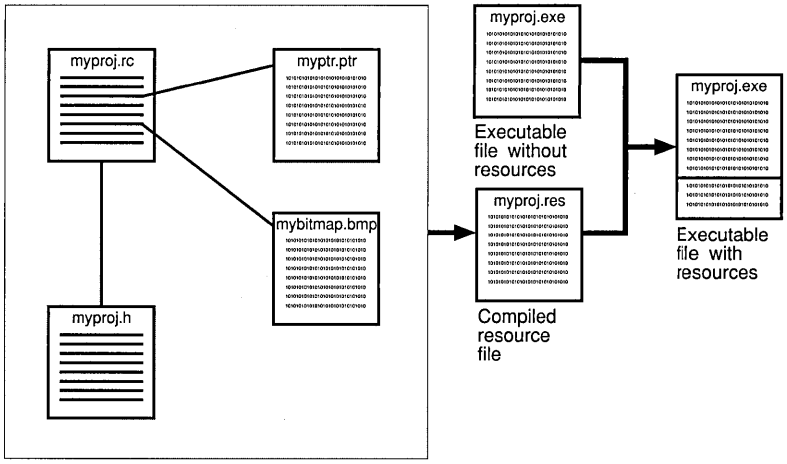
The following figure shows a diagram of these file connections:

Figure 3.8
MYPROJ.RC points to .PTR, .BMP, and .H files



When you've completed the project file, you can compile it into a single .RES file with Resource Workshop and then bind your resources to an existing executable file with your Borland C++ compiler.

Figure 3.9
MYPROJ.RC bound into executable file



Most Resource Workshop projects are considerably larger and more complex than this simple example, and there are other avenues you can take. For example, you can skip the .RES file and compile directly from your source data into an executable file.

However you approach the task, the project remains the central element in creating a resource. The next section describes how Resource Workshop helps you manage the pieces of your project.

Working with resources

This section discusses the following topics:

- Loading a resource
- Choosing a resource editor
- Adding a resource
- Renaming a resource
- Specifying resource memory options
- Removing a resource

Loading a resource

To load a resource, you can do either of the following:

- Double-click the resource name in the Project window.

  Resource Workshop automatically starts the appropriate graphics-oriented resource editor, if one is available. If a resource editor is not available, Resource Workshop starts the internal Script editor.

- Select the resource name in the Project window and then choose either View | Edit Visually or View | Edit as Text.

Always use the resource's name when using the Project window as just described. When the Project window shows resources by file, do *not* use the file name for linked resources (doing so will have no effect).

Choosing a resource editor

When you load a resource, Resource Workshop opens an editor for the resource.

- If you double-click a dialog box, Resource Workshop opens the Dialog editor.
- If you double-click an icon, pointer, or bitmap, Resource Workshop opens the Paint editor.
- If you double-click any other type of resource, Resource Workshop opens the internal text editor.

Using the internal Script editor

To create menus, accelerators, string tables, user-defined resources, and several other resource types, you must use the internal text editor. In addition, you can edit any resource script by selecting the resource in the Project window and choosing View | Edit As Text.

If the resource is in binary format (like a dialog box), Resource Workshop decompiles it to let you work with the resource script.

See Chapter 6 and the Help index for a description of the resource script language.

The internal text editor is a simple ASCII text editor. It uses the *Del, Home, End, PgUp, PgDn, Tab,* and *BkSp* keys to move the cursor and edit text. You can toggle between insert mode and overwrite mode by pressing the *Ins* key.

When you enter text, don't spend any time formatting it, because Resource Workshop is likely to rearrange the text for you when it compiles the resource.

When you're finished making changes, close the text editor. Resource Workshop then asks if you want to compile. (You can also compile the resource at any time by choosing Resource | Compile, which is available only when you are in the text editor.)

You must click OK to save your changes. If you click No, Resource Workshop discards any changes you've made.

If Resource Workshop finds any errors, it tells you what they are and puts you back in the text editor so you can correct them.

If you want to edit the resource script directly without the assistance of Resource Workshop, you can open the source file with an editor of your

choice and edit that file, but Resource Workshop will delete any comments you add the next time you open the file.

When Resource Workshop loads a project, it recompiles the file and reformats the resource script. If it encounters any comments in the script when you open the resource for editing, it displays a dialog box warning you that the comments will be deleted.

Adding resources

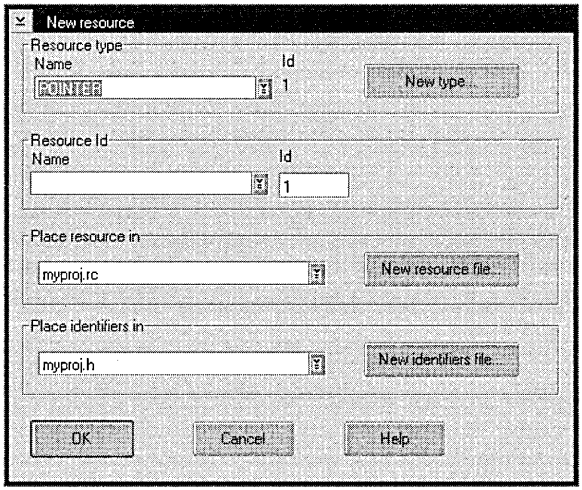
You can add a resource by embedding it in the project, or you can link it as a file reference. To embed a resource, use the New Resource dialog box (Figure 3.10); to link a file, use the Add File to Project dialog box (Figure 3.11).

Embedding resources: New Resource dialog box

To embed a new resource in your project, open the project and display the New Resource dialog box by choosing Resource | New.

If you're viewing the Project window by type, you can also display the New Resource dialog box by double-clicking the name of the resource type you want from the Project window's list of available resource types.

Figure 3.10
New Resource dialog box



1. If the resource type you want isn't already in the Resource Type list box, scroll the list and select it.

   To create a new, user-defined resource type, click the New button to the right of the Resource Type list box.

2. Resource Workshop automatically supplies a numerical Resource ID value. A Resource ID is required, but you can change the value if, for example, you want all your dialog boxes to be in a sequence starting with number 400.

Identifiers are described in detail beginning on page 38.

To create an identifier, enter a name for your resource in the Resource ID field. (The name is optional, but names like dlg_OpenFile makes it much easier for you to identify your resources.)

3. By default, the resource will be added to the current project file, whose name appears in the Place Resource In list box. To place the resource in another project, choose its file name from the list box.

   At this point, you can elect to place the new resource in an external file. If you click the New RC File button, Resource Workshop displays the Add File to Project dialog box. You can save the resource to a new .RC or .DLG file that is automatically linked to the current project file.
4. If you've already created a header file for your identifiers, specify it in the Place Identifiers In list box. Otherwise, click the New Ident File button to display the Add File to Project dialog box, and create a new .H file.
5. When you've completed the New Resource dialog box—resource type, resource ID, the project file to store the resource in, and the header file for the project's identifiers—click OK to exit the dialog box.

When you exit the New Resource dialog box, Resource Workshop starts the appropriate resource editor: the Dialog editor, Paint editor, or internal text editor.

Linking resources: Add File to Project dialog box

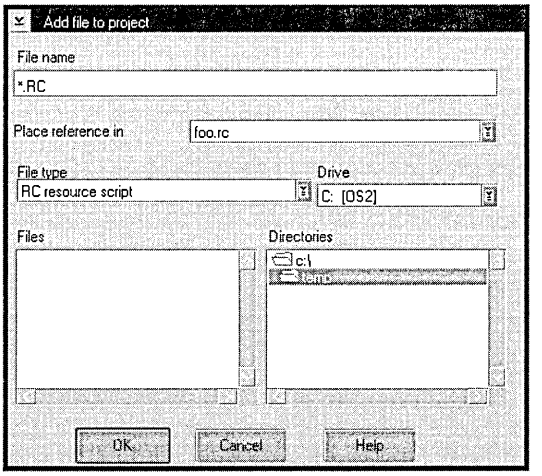
To link a resource stored in an external file to the current project, use the Add File to Project dialog box. You can use this dialog box to add an existing resource file or to create a new file for a new resource.

Existing resource files

This section describes how you link an existing resource to your project. For example, you might have a pointer stored in a .PTR file that you want to use in the current project.

1. Open the project to which you want to link the resource.
2. Choose File | Add to Project. Resource Workshop displays the Add File to Project dialog box.

Figure 3.11
Add File to Project dialog box



3. Type the name of the resource file in the File Name field, or double-click the file name if it's listed in the Files list box.

   If the file isn't in the current directory or is of a different type from the current type, you can select it in either of these ways:

   - You can type the file's full path and name into the File Name field. For example, you might type `D:\ICONS\LOGO.ICO`.
   - You can change drives and directories using the Drive and Directories list boxes. To restrict the files listed in the Files box, use wildcards in the File Name field (for example, *.PTR for pointer files) or select the type from the File Type list box. Then double-click the file name you want in the Files list box.

4. Make sure the name of the file to which you want to link the resource appears in the Place Reference In field. If you want to put the resource in another file, scroll the list to find the name of the file you want.
5. Press *Enter* or click OK to add the file to the project. Resource Workshop puts an entry that points to this file in the Project window.

   If you choose View | By File, you'll see the file name listed and under it the resource name. Any changes you make to this resource are saved to the external resource file.

New resource files

To create a new resource and link it to a project,

1. Open the target project.

2. Choose File | Add to Project to display the Add File to Project dialog box.
3. Enter the name of the file you want to create in the File Name field. If you don't use one of the standard resource file extensions (like .PTR for pointers), use the File Type list box to identify the type of resource you're creating.
4. Make sure the name of the project to which you want to link the resource appears in the Place Reference In field. If you want to put the resource in another project, scroll the list to find the name of the file you want.
5. Press *Enter* or click OK to exit the Add File to Project dialog box.
6. Resource Workshop tells you that the file you've named doesn't exist and asks if you want to create it. Click OK.

Resource Workshop creates a file of the specified type (based on the file extension or, if the extension isn't standard, the file type), inserts a reference to the file in the Project window, and starts the appropriate editor.

Renaming a resource

To rename a resource, choose Resource | Rename. Resource Workshop displays the Rename Resource dialog box.

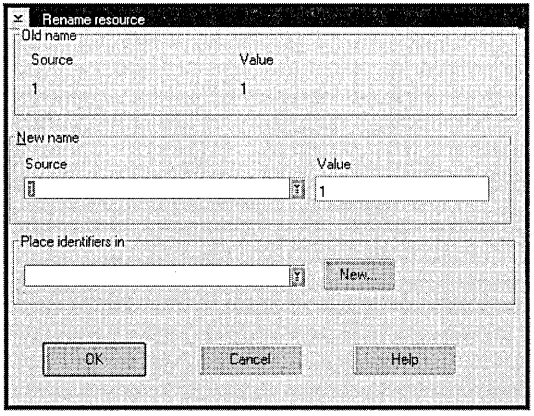


Figure 3.12
Rename Resource dialog box

In the New Name entry field, type the new resource name. If you want to change the ID value, enter a new value in the field provided.

Specifying memory options

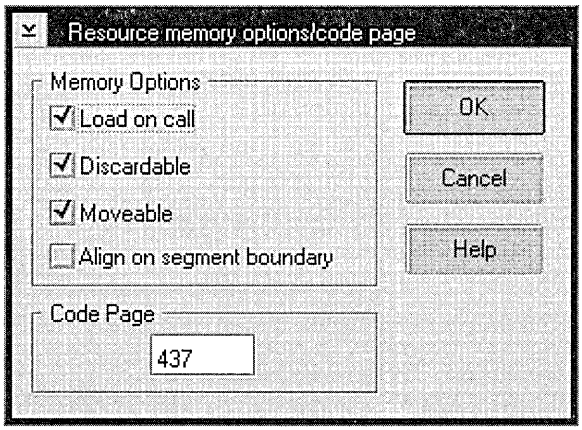
Resource Workshop lets you specify how each resource in your project is loaded and managed in memory. You can also specify the *code page*, which identifies the character set used in the resource.

To specify load and memory options, select the resource in the Project window and then choose Resource | Memory Options to display the

Resource scripts are described in Chapter 6.

Resource Memory Options/Code Page dialog box. If you're writing a resource script, you specify load and memory options (where applicable) as part of the script.

Figure 3.13
Resource Memory Options/Code Page dialog box



Uncheck any options you don't want. For some bitmapped resources, you might want to uncheck the Discardable option. If this option is unchecked, the application can modify the resource while it's in memory.

If you set memory options for a bitmapped resource, those options apply to all the images in that resource.

Table 3.2
Resource load and memory options

Option	Description
Load on Call	Loads the resource into memory only when it's needed. Choosing Load On Call can reduce the amount of time required to load your program.
Discardable	Lets OS/2 discard the resource segment from memory when it's no longer needed. OS/2 can load the resource into memory again when necessary.
Moveable	Lets OS/2 move the resource segment in memory to make room for other memory allocations.
Align On Segment Boundary	Forces OS/2 to align the resource's memory on a segment boundary.

Table 3.3
Resource code page options

Option	Description
437	United States
850	Multilingual
860	Portuguese
863	Canadian French
932	Japanese

Removing a resource

To delete a resource from a project, select the resource in the Project window, then choose either Edit | Cut or Edit | Delete to remove it. (Edit | Cut lets you paste the resource elsewhere.)

Using identifiers

OS/2 requires that every resource be associated with a unique integer called a *resource ID*. By default, Resource Workshop assigns a number to each new resource.

The default number isn't very descriptive, so Resource Workshop lets you assign the resource an *identifier* (a C **#define**) that consists of two parts: a text literal (the identifier name) and the integer value or constant expression. For example, the following statement declares an identifier:

```
#define dlg_OpenFile 100
```

In this example, the identifier's name is dlg_OpenFile and its value is 100. Resource Workshop lists the resource in the Project window as DIALOG: dlg_OpenFile, which readily identifies it as the application's Open File dialog box.

Identifiers must be unique within a resource type. Only the first 31 characters are significant; Resource Workshop ignores any characters past the 31st character.

If you're working with a .RES file, an executable file, or a DLL, Resource Workshop decompiles all resource IDs in the file into integer values. You can't add identifiers to this type of file, but you can save the file as an .RC file and then assign identifiers to its resources. See the section "Working with binary files" on page 28.

Identifier files

When you open a new project, the first thing you should do is specify one or more header (.H) files in which to store your identifiers. These header files use **#define**s to assign values to the identifier names.

This manual refers to header files as *identifier files*.

You can use a text editor or word processor to create your identifier files, but you can do it with Resource Workshop almost automatically, as described in the next section and in the section beginning on page 40.

Creating identifier files

After you open a new project (File | New Project) and give it a name (File | Save Project), add the identifier file by taking these steps:

1. Choose File | Add to Project. Resource Workshop displays the Add File to Project dialog box.
2. Click the File Type list box down-arrow button to display a list of file types you can add to your project.
3. Choose the following line:

   ```
   H C/C++ include file
   ```

4. In the File Name field, type a name for the identifier file.
5. Click OK to exit the Add File to Project dialog box. Resource Workshop creates the identifier file at this time.

C header files

As noted previously, the **#define**s in the header file assign integer values to the identifier names.

The following is a sample from a typical header file:

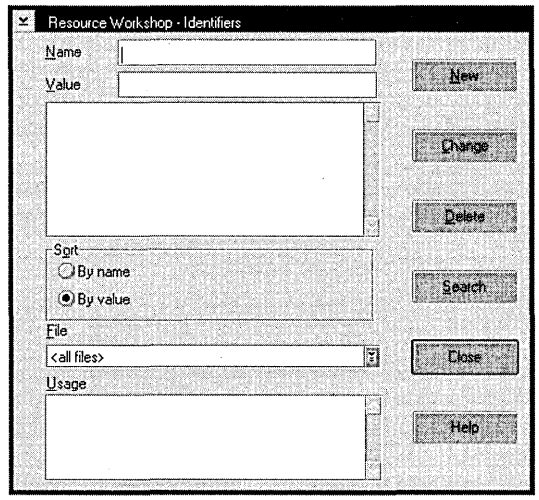
```
/***************************************************
 *     Selected #defines from RWCDEMOC.H           *
 ***************************************************/

#define  bmp_StatusBar              101
#define  cm_About_CUA               145
#define  id_ClearWindow             229
#define  dlg_About                  104
#define  dlg_FileNew                106
#define  sth_Edit                    15
#define  men_Main                   100
#define  acc_Main                   100
#define  ico_RWCDemo                100
#define  sth_EditClear               13
#define  ScribbleWindow             100
#define  FileWindow                 101
#define  GraphWindow                102
```

In addition to **#define**s, you can also store type and structure definitions, program code, and comments in a header file. Resource Workshop ignores all data in the header file except for the **#define**s and any preprocessor directives.

Using the Identifiers window

You can use the Identifiers window to assign, edit, delete, or list identifiers and to start a resource editor. To display the Identifiers window, choose View | Identifiers Window from the Project window or from any editor window.

Figure 3.14
Identifiers window



The Identifiers window is a *modeless* dialog box, which means you can leave it open as you work in an editor or perform other functions. If you want it out of the way, you can close the window or minimize it.

The Identifiers window has the following components:

- The Name field shows the currently selected identifier.
- The Value field shows the ID value associated with the selected identifier.
- The list box below the Name and Value fields (we'll call it the Identifiers list box) lists the identifiers in the file named in the list box immediately below. You select identifiers in this list box.
- The Usage list box shows the type and name of the resource whose identifier is highlighted in the Identifiers list box. If the highlighted identifier is not associated with a resource, the Usage box says "(unused)."

Assigning identifiers

You'll typically assign identifiers to your resources at the time you create them, but you can assign identifiers at any time, even before you create the associated resources.

To assign an identifier to a resource as you create it, you use the New Resource dialog box, as explained on page 33.

To assign an identifier before you create the resource,

1. Open the Identifiers window (View | Identifiers Window).
2. Use the File list box to specify the file in which the identifier is to be stored.

3. Type the resource name in the Name field.
4. Type the ID value in the Value field.
5. Click New.

   Note that the new resource name now appears in the Identifiers list box, and that its Value is given as "(unused)."

When you create a resource and give it a name to which you've already assigned an identifier, the resource is automatically associated with that identifier.

For example, you might create an identifier named CHECK_DIAL with the ID value 1200. If you create a new dialog box and use Resource | Rename to give it the name CHECK_DIAL, the dialog box is automatically associated with the ID value 1200. You can verify this by looking at the Identifiers dialog box or by saving the project and then looking at the identifiers file.

To assign an identifier to an existing resource,

1. Select the resource in the Project window.
2. Choose Resource | Rename. Resource Workshop displays the Rename Resource dialog box.
3. Type the identifier you want to assign in the New Name field. If this is a new identifier, you must also type its value in the Value field.

Editing identifiers

To change an identifier's value, do the following:

1. Choose View | Identifiers Window to display the Identifiers dialog box.
2. Select the identifier whose value you want to change.
3. Click the Change button. Resource Workshop displays the Change Identifier Value dialog box.
4. Type a new value in the New Value field and click OK.

The new identifier value will be written to your .H file the next time you choose File | Save Project.

Deleting identifiers

If an identifier is not used in your project, you should delete it from the .H file. Here are three reasons you might have an unused identifier:

- You assign an identifier to a resource and then delete the resource.
- You add an identifier to the project and then never use it.
- You rename a resource that already has an integer identifier value.

To delete an identifier,

1. Choose View | Identifiers Window to display the Identifiers dialog box.
2. Select the identifier you want to delete.

   If the selected identifier is not associated with a resource (either because the resource was deleted or the identifier was never used), the Usage box says (unused).

   You *can* delete an identifier that is still in use.

   If, however, the identifier is still associated with a resource, the Usage box automatically highlights the type and name of the associated resource.
3. Click the Delete button.

   If the identifier is unused, it is deleted immediately. No warning dialog box is displayed.

   If the identifier is still in use, Resource Workshop displays a warning dialog box that says "#define is used. Delete anyway?" To delete the identifier, click the Yes button. If you don't want to delete the identifier, click the No button.
4. The next time you choose File | Save Project, Resource Workshop updates the identifier file, removing the deleted identifier.

Listing identifiers

To list the identifiers in your project, do the following:

1. Choose View | Identifiers Window to display the Identifiers dialog box.
2. Use the File combo box to select the identifier file whose identifiers you want to see.
3. Scroll the Identifiers list box to the identifiers you want to see. When you highlight an identifier in the list box, its name and integer value appear in the Name and Value boxes above the list box.

Starting a resource editor

You can use the Identifiers dialog box to start a resource editor with a preselected resource already loaded.

To start a resource editor from the Identifiers dialog box,

1. Scroll the Identifiers list box until the resource you want is highlighted.

   The resource's type and name appear in the Usage list box.
2. Double-click on the highlighted type and name in the Usage list box.

   Resource Workshop starts the appropriate editor with that resource already loaded.

C H A P T E R 4

Using the Dialog editor

Using Resource Workshop's Dialog editor, you can create and modify the templates for dialog boxes and PM windows. This chapter explains how to use the Dialog editor and its tools. It covers the following topics:

- Understanding dialog resources
- Using the Dialog editor
- Using the Dialog editor tools
- Working with controls in general
- Working with specific controls

Understanding dialog resources

Dialog resources provide templates for dialog boxes and windows in your PM programs. That is, they describe the visual appearance of dialog boxes without defining their actual functions. Dialog boxes generally contain *controls*, such as push buttons, text boxes, and scroll bars. Controls enable the user to specify information or choose among options, but they can also display static text and graphics. Figure 4.1 shows a typical PM dialog box with several controls.

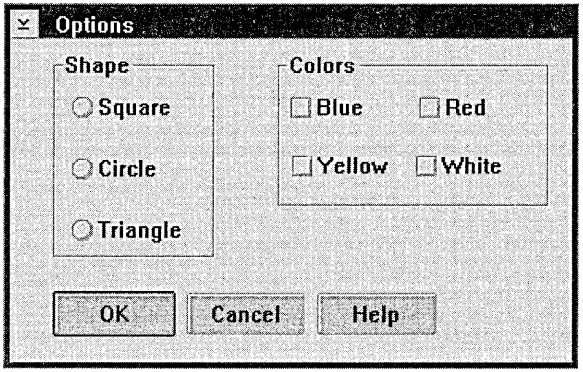


Figure 4.1
A typical PM dialog box

What a dialog resource does

At its simplest, a dialog resource specifies the initial position, size, style, and resource ID of a dialog box. The real power of dialog resources, however, lies in the ability to specify the type and location of the controls in a dialog box. Instead of writing code to create and position each control, the dialog resource gives your program a complete description of the dialog box and its controls.

Using the Dialog editor

There are a number of capabilities in the Dialog editor that you need to understand, regardless of the type of work you'll be doing in the editor. This section explains the following tasks:

- Starting the Dialog editor
- Setting dialog box attributes
- Moving the dialog box
- Undoing changes
- Testing your dialog box
- Saving your work
- Customizing the Dialog editor
- Using a control grid
- Setting presentation parameters

Figure 4.2 shows a typical Dialog editor window with its parts labeled.

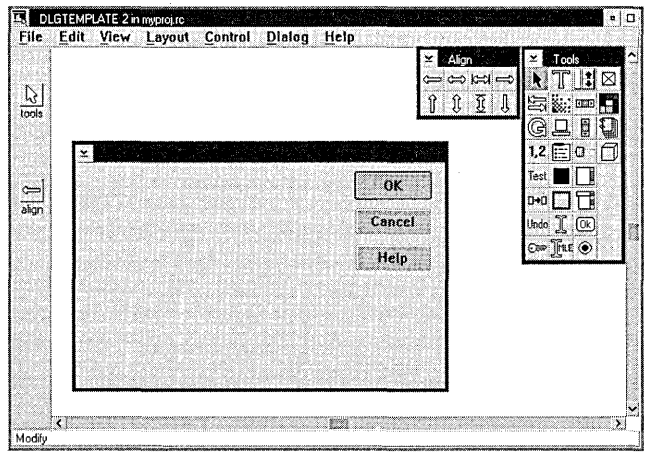


Figure 4.2
The Dialog editor with an empty dialog box

Starting the Dialog editor

The Dialog editor can either create a new dialog resource or edit an existing one. In either case, you must have a project open. If you need information on projects or the project window, see Chapter 2, "Resource Workshop basics."

Editing an existing dialog box

From the project window, locate the dialog box you want to edit and double-click it to start the Dialog editor. You can also select the desired dialog box with the mouse or the keyboard then choose either View | Edit Visually or View | Edit As Text to start the Dialog editor or the Script editor, respectively.

Creating a new dialog box

To start the Dialog editor and create a new dialog resource, you can do either of two things from the project window:

- Double-click the word DLGTEMPLATE. The New Resource dialog box appears with DLGTEMPLATE already selected in the Resource Type combo box. Click OK. Resource Workshop assigns an automatic ID and stores the resource in the main project file.
- Choose Resource | New. The New Resource dialog box appears. Choose DLGTEMPLATE in the Resource Type combo box and click OK. You can also assign an identifier and a different file for this resource if you want.

Before starting the Dialog editor, Resource Workshop shows you the New Dialog dialog box, which lets you choose one of three starting templates for your new dialog box:

- A standard dialog box with OK, Cancel, and Help buttons down its right side
- A standard dialog box with OK, Cancel, and Help buttons along the bottom
- A standard window

You can also add a caption to the title bar of the new dialog box in the Caption field of the New Dialog dialog box.

When you click OK in the New Dialog dialog box, Resource Workshop starts the Dialog editor with your chosen dialog box ready to edit, as shown in Figure 4.2.

Setting dialog box attributes

You can change most of the attributes of a dialog box through its Dialog Style dialog box. To display the Dialog Style dialog box, you can either double-click the dialog box's caption bar or select the dialog box and choose Control | Style.

In the Dialog Style dialog box, you can set a number of different attributes for your dialog resource, as described in the following sections.

Setting dialog styles

Dialog boxes, like all PM windows, have style flags that control their appearance and behavior. Within the Dialog style dialog box you'll find check boxes for each of the available styles. The following table summarizes the styles and their effects.

Dialog style	Meaning
Visible	The dialog box is initially visible.
Disabled	The dialog box is initially disabled.
Save Bits	Saves the area covered by the dialog box as a bitmap if sufficient memory is available. Without this style, when the dialog box is destroyed, the covered area is restored by invalidating the appropriate areas in the windows being uncovered.
Sync Paint	The window repaints synchronously, meaning the window gets a WM_PAINT message whenever any region is invalidated. Without this style, PM batches paint operations, delaying the WM_PAINT until all nonpaint messages leave the thread's queue.
System Modal	No other window or process is active while the dialog box is displayed.
No Byte Align	Prevents PM from forcing the dialog box to align on a pel grid with its origin on a byte boundary. Byte alignment can improve the speed of background repainting.
No Move With Owner	The dialog box keeps its position when its owner window moves.
DBCS Status Line	Adds a DBCS-compatible status line to the bottom of the dialog box.

Setting border styles

You can select any of four border styles for your dialog box. The following table summarizes the different borders.

Border option	Description
Dialog	Standard dialog box border
Thin	Narrower than the standard border
Sizing	Similar to the standard border, but the user can drag the border to resize the dialog box
None	No border at all

Setting frame controls

The Dialog Style dialog box gives you the ability to choose a number of controls for the frame of your dialog box. Checking an option includes the specified control on the edge of the dialog box. The following table summarizes the choices.

Control option	Meaning
Title Bar	Title bar across the top of the dialog box
Minimize button	Button to shrink the dialog box to an icon
Maximize button	Button to zoom the dialog box to full screen size
System Menu	System menu box in top left corner of the dialog box
Vertical Scroll Bar	Scroll bar on right side of the dialog box
Horizontal Scroll Bar	Scroll bar across bottom of the dialog box

Setting window alignment

The initial position of your dialog box can be set relative to any of three items: the origin of its parent window, the origin of the screen, or the position of the mouse. You can choose any of these options in the Dialog Style dialog box.

Setting memory flags

You can set the memory options for the dialog resource by choosing Dialog | Memory Options. This displays the Dialog Memory Options dialog box, where you can set any or all of the Preload, Moveable, and Discardable flags for the resource memory.

Moving the dialog box

The position of the dialog box in the editor window reflects the position the dialog box will assume when your application displays it. You can change that initial position by clicking the title bar of the dialog box and dragging it to the desired position. You can also set the size and position of the dialog box directly by selecting the dialog box and choosing Layout | Size And Position.

Undoing changes

You can "undo" any editing you do in the Dialog editor, such as placing controls, aligning them, deleting them, and so on, with the Undo tool or the Edit | Undo menu command. Undo works on commands that affect groups of controls as well as commands that change single controls.

Testing your dialog box

To test your dialog box to see the effect of any changes you've made, select the Test tool or choose Options | Test Dialog. This makes a "live" copy of the dialog box you're editing.

When testing, you can press *Tab* and the arrow keys to see how you can move around your dialog box, or you can type text to see how text controls handle scrolling. Check to see if your controls are in the order you want them.

While testing a dialog box, the status line at the bottom of the Dialog editor reads `Test`.

To leave test mode and return to edit mode, do any of the following:

- Click OK or Cancel in the dialog box.
- Choose Options | Test Dialog again.
- Press *Enter*.
- Click the Selector bitmap in the Tools palette twice. (The first click switches focus from your dialog box to the Dialog editor.)

Saving your work

It's a good idea to save your changes as you go along, rather than waiting for Resource Workshop to prompt you when you close the project. To save your dialog box, choose File | Update or File | Save All in the Dialog editor or File | Save Project in the project window.

Resource Workshop holds a memory image of the project's resources while you work. File | Update updates the memory image only, which enables you to test interactions among resources, such as a dialog box that uses a bitmap from another resource in the project. File | Save All saves the entire project and updates all disk files.

Customizing the Dialog editor

You can set several options for the Dialog editor. Choose File | Preferences to display the Dialog Editor Options dialog box.

Setting selection options

The Selection Type group enables you to customize the selection of multiple controls with the mouse. If you check Select Near Border, Resource Workshop forces you to select controls by clicking inside their boundaries, within four pels of the border. If you check Selection Rectangle Surrounds, the selection rectangle must completely surround a control in order to select the control.

Both options are unchecked by default.

Setting units options

The status line can display position and size information for controls in either dialog units or screen units. By default, Status Line Units is set to Dialog.

Setting border options

You can set the selection border to either a thick line or a thin one. The default setting is a thick selection border.

Using a control grid

One way to make sure the controls in your dialog boxes line up correctly is to use a *grid*. A grid is a set of horizontal and vertical lines you can use to position your controls. The grid affects only the controls you add. Controls you've already placed don't move if you change the grid.

Displaying the grid

By default, the Dialog editor's grid is hidden. To make it visible, choose Layout | Grid Settings, and check the Draw Grid box.

Snapping to the grid

You can force your controls to align themselves to the Dialog editor grid as you add them by choosing the Layout | Snap To Grid menu command. When you choose the Snap To Grid option, all controls you add automatically move to the nearest point on the grid, even if the grid is not visible.

Moving existing controls to a grid

To force controls already in the dialog box to align with the dialog box's grid, select the controls you want to align and choose Layout | Force Alignment. All selected controls will move to the nearest point on the grid.

Changing the grid

To change the horizontal or vertical spacing of the control grid, choose Layout | Grid Settings. You can type new values for the horizontal (X) and vertical (Y) sizes of the grid.

Setting presentation parameters

All PM windows, including dialog boxes and their controls, have a set of properties called *presentation parameters*, including font, color, color index, and custom information. The most common use of presentation parameters in dialog boxes is to change fonts.

PM windows inherit presentation parameters from their owners. Thus, if you don't set specific presentation parameters for a control, it uses those of the dialog box that owns it. You can therefore make changes that apply to all controls in a dialog box by changing the presentation parameters of the dialog box itself.

To set presentation parameters for a dialog box or control,

1. Select the item you want to change presentation parameters for.
2. Choose Control | Presentation Parameters, which displays the Presentation Parameters dialog box.

The Presentation Parameters dialog box shows any currently defined presentation parameters for that item.

3. To change or delete an existing presentation parameter, select the parameter in the Current Presentation Parameters list box and click either Change or Delete.

   To define a new presentation parameter, click the appropriate button in the New group for the kind of presentation parameter you want to add. Each displays an appropriate dialog box where you can enter a new presentation parameter.
4. When you finish setting presentation parameters, click OK to close the Presentation Parameters dialog box.

Using the Dialog editor tools

The Dialog editor has two floating palette windows that contain square gray bitmapped images that represent *tools*. Clicking on one of the tools is a shortcut for choosing a menu item and can greatly speed your editing.

Overview of the tools

The Dialog editor has three kinds of tools:

- Mode tools
- Action tools
- Control tools

The next sections describe each kind of tool and the Dialog editor's sidebar.

Using mode tools

Mode tools set the Dialog editor's current operating mode. Depending on the mode you select, the mouse pointer has different effects on the controls in the dialog box. Each of the mode tools corresponds to one of the commands on the Dialog menu. Table 4.1 shows the mode tools, their equivalent menu commands, and the actions associated with them.

Table 4.1
Dialog editor mode tools

Tool	Menu command	Action
	Dialog|Modify Controls	Selects a control to modify.
	Dialog|Set Tab Stops	Toggles tab stop at each control clicked.
	Dialog|Set Order	Sets tab order of controls in the order you click them.
	Dialog|Set Groups	Toggles clicked control as first control in a group.
	Dialog|Test	Toggles test mode.

Using action tools

Action tools perform a specific action when you click them. For example, all the tools on the Align palette are action tools. When you click one, it immediately aligns the selected controls in the dialog box. Each of the tools on the Align palette corresponds to an option in the Align Controls dialog box.

There are also several action tools on the Tools palette. These tools and their menu equivalents and use are summarized in Table 4.2.

Table 4.2
Dialog editor Tools palette action tools

Tool	Menu command	Action
	Edit|Duplicate or Align|Array	Copies the selected control or creates an array of controls. If a single control is selected, this tool copies it. If multiple controls are selected, this tool aligns them as an array.
	Edit|Undo or Edit|Redo	Restores the dialog box to its state before the last action. When used with *Shift*, performs Redo instead of Undo.

Using control tools

Control tools enable you to select a control to add to the dialog box. Clicking a control tool is equivalent to choosing a control type from the Control menu or choosing a control type from the New Control dialog box.

Using the sidebar

The Dialog editor window's sidebar contains an alignment indicator that shows the most recently used alignment tool and a tool indicator that shows the currently selected tool. You can drag the sidebar to either side of the window.

If you've closed either the Align palette or the Tools palette, you can reopen them by clicking the sidebar's alignment indicator or tool indicator, respectively.

Working with controls in general

Most of the work you do in the Dialog editor involves the creation, placement, and modification of *controls*. Controls are the items appearing inside the dialog box, including push buttons, list boxes, and entry field controls. Most of the tasks you perform on controls are the same for each type of control, so this section explains the common tasks such as creating, moving, resizing, and modifying. The next section explains any specific information for the individual control types.

No matter what kind of control you're working with, certain tasks work exactly the same way. These tasks are

- Creating the control
- Moving the control
- Resizing the control
- Aligning controls
- Arranging controls
- Setting control attributes
- Setting tab stops
- Changing tab order
- Grouping controls

This section describes these common operations. The following sections tell you specific information unique to each kind of control. Tasks that involve multiple controls, such as aligning and grouping, are explained in the section on using the Dialog editor tools, starting on page 50.

Creating a control

Creating a new control in a dialog box takes two steps:

- Picking a control type
- Placing the control in the dialog box

Picking a control type

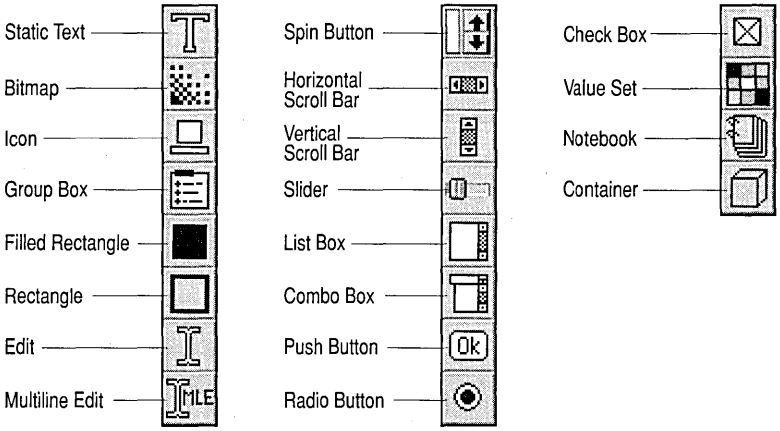
There are two ways to pick the type of control you want to create:

- Pick a control from the Tools palette
- Use the Control | New menu command and the New Control dialog box

Picking controls from the Tools palette is generally easier than using the menu and dialog box, but using the dialog box enables you to see samples of the controls before inserting them.

To pick a control from the Tools palette, click the left mouse button on any of the controls pictured in the palette. Figure 4.3 shows which control each bitmap represents.

Figure 4.3
The control bitmaps on the Tools palette



To pick a control using menus and dialog boxes, choose Control | New, which displays the New Control dialog box. From the Control Type combo box, pick the type of control you want to create. The Sample area shows an example of the control you pick. When you've picked the control you want, click OK.

Whichever method you use to pick the control type, the pointer assumes the shape of the control, so you can tell what kind of control you're creating. The pointer image is similar to the bitmap on the palette.

Placing the control

Once you've picked a control type, your mouse pointer assumes an image that represents the control you're creating. You can then move the mouse anywhere in the dialog box you're editing. When you click the left mouse button, the control appears in the dialog box with its upper left corner at the point where you clicked. If you have the Snap To Grid option turned on, the control appears with its upper left corner at the grid point nearest where you clicked.

You don't have to worry too much about where you first place the control, as you can easily move or resize it later.

Selecting controls

You can select a control to modify using either the mouse or the keyboard. The selected control has a thick border around it.

- To select a control with the mouse, click the control with the left mouse button.

- To select a control with the keyboard, press *Tab* to cycle through the controls until the control you want is selected. You can cycle in the opposite order by using *Shift+Tab*.

Selecting multiple controls

You can select multiple controls by holding down *Shift* while you click on controls. The thick border indicating selection enlarges to surround all the selected controls. Selecting with *Shift*+click enables you to select only specific controls.

You can also select a group of adjacent controls by clicking the selection tool on the dialog box, not touching any control, and then dragging the pointer. As you drag, Resource Workshop draws a *selection rectangle* (sometimes called a "rubber band") from the point where you originally clicked to the point under the pointer's hot spot. When you release the mouse button, the Dialog editor selects any controls inside the selection rectangle.

You can customize the behavior of the selection rectangle. One of the options in the Dialog editor preferences dialog box lets you choose whether dragging selects only those controls entirely within the selection rectangle or all the controls contained or touched by the selection rectangle. See "Setting selection options" on page 48.

Moving a control

You can move a control using either the mouse or the keyboard.

- To move a control with the mouse, make sure you're using the Modify tool. Click on the control you want to move and drag it to the desired position.
- To move a control with the keyboard, select a control and press an arrow key to start moving the control. The selection rectangle moves to indicate the new position of the control. You can then press other arrow keys until you get the control where you want it. Press *Esc* or *Enter* to stop moving the control.

Resizing a control

You can resize a control using either the mouse or the keyboard.

- To resize a control with the mouse, click the thick border around the selected control and drag the border to the desired position. When you release the mouse button, the control assumes the size of the thick border.
- To resize a control with the keyboard, hold down *Ctrl* and press an arrow key to move the corresponding side of the control. For example, pressing *Ctrl*+↑ moves the top of the control up, *Ctrl*+→ moves the right side to the right, and so on.

Holding down *Shift* along with *Ctrl* resizes in the opposite direction, so for example, *Ctrl+Shift+*↑ moves the bottom of a control up.

You can also set the exact size of a control by selecting it and choosing Layout | Size And Position, which displays a dialog box in which you can specify the height (CX) and width (CY) of the control.

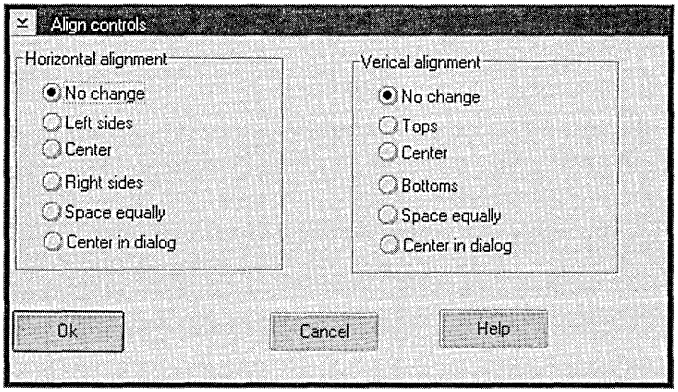
Aligning controls

Once you've added controls to your dialog box, Resource Workshop gives you several ways to align them.

Before you can align, resize, or arrange multiple controls, you must select the controls you want to adjust. See "Selecting controls" on page 53. You can also align single controls.

Once you select the controls to align, you can use either the menu system or the Alignment palette to align the controls. To use the menus, choose Layout | Alignment, which displays the Align Controls dialog box shown in Figure 4.4.

Figure 4.4
Align Controls dialog box



All the options in the Align Controls dialog box have equivalents on the Alignment palette. They also have keyboard accelerators. Table 4.3 shows the horizontal alignment commands. Table 4.4 shows their vertical counterparts.

Table 4.3: Horizontal alignment options

Palette	Keyboard	Dialog box	Description
		No Change	There is no change in horizontal alignment.
	*Ctrl+L*	Left Sides	The left sides of all controls move to the left side of the selection frame.
	*Ctrl+H*	Centers	The horizontal centers of the controls move to the center of the selection frame.
	*Ctrl+R*	Right Sides	The right sides of the controls move to the right side of the selection frame.
	Ctrl+Stretch*	Space Equally	Moves the controls horizontally within the selection frame so the spaces between them are equal.
	*Ctrl+D*	Center in Dialog	Moves the selection frame horizontally so it's centered in the dialog box. The relative positions of the individual controls within the selection frame is unchanged.

* *Ctrl*+Stretch means to hold down *Ctrl* while dragging the selection frame around multiple controls with the mouse.

Table 4.4: Vertical alignment options

Palette	Keyboard	Dialog box	Description
		No Change	There is no change in vertical alignment.
	*Ctrl+T*	Tops	The tops of all controls move to the top of the selection frame.
	*Ctrl+V*	Centers	The vertical centers of the controls move to the center of the selection frame.
	*Ctrl+B*	Bottoms	The bottoms of the controls move to the bottom of the selection frame.
	Ctrl+Stretch*	Space Equally	Moves the controls vertically within the selection frame so the spaces between them are equal.
	*Ctrl+C*	Center in Dialog	Moves the selection frame vertically so it's centered in the dialog box. The relative positions of the individual controls within the selection frame is unchanged.

* *Ctrl*+Stretch means to hold down *Ctrl* while dragging the selection frame around multiple controls with the mouse.

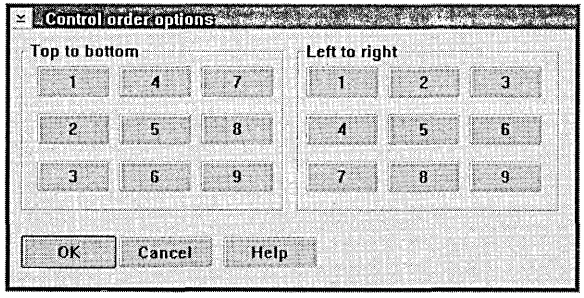
Arranging controls

Resource Workshop enables you to automatically arrange a group of selected controls into an array of evenly spaced, equally sized rows and columns, renumbering the controls in sequence.

To arrange controls into an array,

1. Select the controls you want to arrange.
2. Adjust the size and shape of the selection frame to enclose the area you want to fit the array into. For example, if you make the selection frame larger, the Array command will move the controls out to the new boundaries set by the frame.
3. Choose Layout | Array or click the Duplicate tool. You'll see the Form Controls Into An Array dialog box.
4. Under Array Layout, specify the number of rows and columns you want.
5. Under Order, specify your preference for the ordering of the rows and columns. Figure 4.5 shows how the ordering options affect the same group of nine controls.

Figure 4.5
Control order options



Setting control attributes

To change the attributes of a control,

1. Select the control.
2. Either double-click the left mouse button or press *Enter*. A style dialog box appears.

The style dialog box enables you to change all the attributes of the control.

Setting basic attributes

Each control type has its own unique attributes you can adjust, but they all have four attributes in common, called *basic attributes*. The following table lists the four basic attributes and what they mean.

Attribute	Meaning if checked
Visible	The control is initially visible; otherwise, hidden.
Disabled	The control is initially disabled; otherwise, enabled.
Group	The control is the first in a logical group.
Tab Stop	The user can press *Tab* to select the control.

You can also change the caption and control ID of a control within the style dialog box. The Header File combo box specifies where Resource Workshop looks for identifiers for the Control ID: predefined values, all header files, or both.

Assigning control IDs

Control IDs are the numbers used by your application to identify particular controls in a dialog box. Resource Workshop supports assignment of IDs either with numbers or symbolic constants called identifiers. The identifiers used for control IDs are identical to those used as resource IDs, as explained in Chapter 3, "Working with projects, resources, and identifiers."

Every control's style dialog box contains fields that enable you to assign either numeric or symbolic IDs to the control. You can use identifiers predefined by OS/2 or identifiers you've already assigned by choosing them from the combo box in the Control ID field. You can also define new identifiers by typing the new identifier name into the combo box and typing a number or valid expression in the entry field control to the right of the combo box. When you save the resource, Resource Workshop saves the new identifier in your project's identifier file.

Setting DBCS support styles

The controls that accept typed input from the user (entry field controls, multiline entry field controls, and combo boxes) have a set of attributes called DBCS Support Styles. These determine how the control supports double-byte character sets. The following table summarizes the four options.

Option	Description
Any	The control accepts characters from any character set.
Mixed	The control accepts mixed single- and double-byte character sets.
SBCS Only	The control accepts only single-byte characters.
DBCS Only	The control accepts only double-byte characters.

Setting control-specific attributes

The style dialog box for each control type contains check boxes or radio buttons that enable you to set all the options supported by that control. The specifics of each dialog box are explained in the next section.

Setting tab stops

When using the keyboard, users typically press *Tab* to move from one control (or group of controls) to another. Only controls with their Tab Stop attribute set get the focus through *Tab*. Some types of controls are automatically defined as tab stops when you add them to a dialog box.

You can also change tab stops in a control's style dialog box.

To set or remove tab stops, use the Tab Set tool or the Set Tab Stops menu command.

1. Click the Tab Set tool or choose Dialog | Set Tab Stops. The pointer changes to the Tab Set symbol. The Dialog editor places a white rectangle over each control, with a letter "T" in any box over a control that is a tab stop.
2. To set a tab stop, click an empty white rectangle.

   To remove a tab stop, click a rectangle containing a "T."
3. When you finish changing tab stops, click the Selector tool or choose Dialog | Modify Controls.

Changing tab order

Tab order is the order in which controls in a dialog box get the focus when the user presses *Tab*. By default, tab order is the order in which you add the controls to the dialog box. You can change the tab order of controls using the Set Order tool or the Set Order menu command.

To change the order of the controls in your dialog box,

1. Select the controls whose order you want to change. If you don't select any controls, you can change the order of all the controls.
2. Click the Set Order tool or choose Dialog | Set Order. The pointer changes to the Set Order symbol, and a white box appears over each control you selected. Each box contains a number indicating the order of the controls.

If you make a mistake, you can "reclick" a control to restore its order. You can backtrack by clicking controls in reverse order.

3. Click the controls you want to assign new order numbers to in the order you want them. When you click a control, its white rectangle turns gray. The prompt at the bottom of the Dialog editor window shows the next order number to be assigned.
4. When you finish assigning new order numbers, click the Selector tool or choose Dialog | Modify Controls.

Grouping controls

You can define *groups* of related controls within your dialog boxes. Grouping enables the user to move among the grouped controls using arrow keys. Groups are usually used only with check boxes and radio buttons. You can let users know that controls are grouped by putting them in a group box.

A group is defined by setting the Group attribute in the first control you want grouped. The dialog box treats all subsequent controls in tab order (the order you added the controls) as part of that group, until it encounters another control with the Group attribute.

Defining groups works exactly like setting tab stops. To define groups of controls,

1. Click the Set Groups tool or choose Dialog | Set Groups. The pointer changes to the Set Groups symbol, and white rectangles appear over all controls. Controls with their Group attribute set have a letter "G" in their rectangle.
2. To define a new group, click the empty rectangle over the first control in the group. Make sure the control after the last one in the group also has a "G" in its rectangle.

   To remove a group, click the rectangle over the first control in the group. Its "G" disappears.
3. When you finish setting groups, click the Selector tool or choose Dialog | Modify Controls.

You can also change Group attributes in a control's style dialog box.

Working with particular controls

In addition to changing the caption, control ID, and basic attributes common to all controls, you can also use a control's style dialog box to set attributes specific to the type of control.

The following sections describe the specific attributes you can set through the style dialog box for each control. For each control, you'll find two kinds of attributes you can change, *styles* and *types*.

Styles are generally individual attributes you can toggle on or off, much like the basic attributes. Examples include the visible border of a push button or word-wrapping in a multiline entry field control. Styles generally correspond to the standard style constants defined in os2.h, and the labels in the style dialog boxes reflect the names of those constants.

Types are options that change something fundamental about the control. For example, a static rectangle control can be one of three types: foreground, background, or halftone. Scroll bars have two types: horizontal and vertical. Like styles, types relate to OS/2 constants, but rather than the simple presence or absence of a particular constant, a type represents a choice among exclusive constants.

Static text controls

In addition to the basic attributes, static text controls have three styles and two types you can set.

The three text styles are Halftone (graying), Word Break (showing only complete words), and Mnemonic style (interpreting ~ as a mnemonic prefix character).

The available type options determine the alignment of the text within the control boundaries. You can align the caption text horizontally or vertically.

The next sections describe several variations on static controls.

Bitmapped static controls

Bitmaps and icons in dialog boxes are static controls with special attributes. Although they have no styles other than the basic attributes, a pair of radio buttons in the style dialog box allows you to toggle between bitmap and icon controls.

Rectangle and frame static controls

Solid rectangles and frames (unfilled rectangles) are simplified static controls with no caption. Their only special attribute determines the color of the rectangle or frame: Foreground (dark gray), Background (light gray), or Halftone (cyan).

Group box controls

Group boxes are static controls used to visually group related items, especially check boxes and radio buttons. Group boxes can have visible borders, and you can set text styles for Halftone and Mnemonic as you would with a static text control.

Entry field controls

In addition to the basic attributes, entry field controls have DBCS support styles, as described on page 58. The following table describes the other entry field control styles you can set.

Style	Description
Auto Tab Stop	When a typed character fills the entry field, focus automatically moves to the next field as if *Tab* had been pressed.
Unreadable	The text in the entry field appears as asterisks.
Autoscroll	The text scrolls right or left to keep the insertion point visible.
Read Only	The entry field ignores characters typed in it.
Margin	Gives the entry field a visible border.

You can also control the alignment of the text within the entry field control by selecting the Justification type: Left, Right, or Center.

Multiline entry fields are not different types of entry field controls. Rather they are a separate class of window, as described in the next section.

Multiline entry field controls

Multiline entry field controls are distinct from single-line entry field controls, rather than merely being a different style of entry field control. They are a different window class. Although they are similar in many ways, multiline entry field control resources have their own, different attributes.

In addition to the basic attributes and DBCS support styles, multiline entry field controls have the same Read Only and Margin styles as regular entry field controls. They also have two unique styles, described in the following table.

Style	Description
Ignore Tab	Causes the entry field to ignore the *Tab* key, allowing the dialog box to move the focus to the next control when the user presses *Tab*.
Word Wrap	The entry field control wraps text to a new line when a word overruns the right side of the visible area.

You can also enable or disable either the horizontal or vertical scroll bars present by default.

Button controls

In addition to the basic attributes, push buttons have the following five styles you can set:

Style	Description
Default	Makes the button the default button, which has a wide border indicating it's the default response if the user presses *Enter*.
Help	Sends a WM_HELP message to the owner window when pressed.
SYSCOMMAND	Pressing the button generates WM_SYSCOMMAND messages instead of WM_COMMAND messages.
No Pointer Focus	Clicking the button notifies the owner window with a BN_CLICKED notification, but does not give the button the focus.
No Border	Removes the default border around the button, making the button look like ordinary text, although it still works as a button.

In addition, you can choose among three button types, meaning the button appears as Text (with the caption text displayed), Bitmap (a bitmap image replaces the text), or Icon (an icon image replaces the text).

Radio button controls

Radio buttons have only two styles in addition to the basic attributes. One is the same No Pointer Focus style used by push buttons. The other is No Cursor Select, which makes the button selectable only with the mouse.

You can also choose between two types of radio buttons, regular and auto. Auto radio buttons automatically fill the dot to the left of the text when pressed and clear the dots of other radio buttons in the same group. With regular radio buttons, your program is responsible for updating the appearance of all the buttons in the group when one is pressed.

Check Box controls

Check boxes have only one style in addition to the basic attributes, and that is the same No Pointer Focus style used by push buttons.

You can also choose among four Check Box Types: regular check box, auto check box, 3-state check box, and auto 3-state check box. A regular check box puts its caption text to the right of a square box, and puts a check mark in the box when selected. A 3-state check box toggles among three states, represented by an empty box, a checked box, and a gray, hatched box. Auto check boxes and auto 3-state check boxes toggle automatically when clicked, whereas regular and 3-state check boxes must be toggled by your program.

List box controls

List boxes have five styles in addition to the basic attributes:

Style	Description
No Adjust Position	Disables the list box's default behavior, which adjusts the size of the displayed control to show only whole lines of text.
Multiple Selection	The user can select more than one item in the list by using *Shift*+click.
Extended Selection	The user can select individual items or ranges of items.
Horizontal Scroll	Enables the user to scroll the list horizontally to read text that exceeds the bounds of the control.
Owner Draw	The application paints the items in the list box.

Combo box controls

In addition to the basic attributes and DBCS support styles, combo boxes have one additional style, the same Horizontal Scroll style used by list boxes.

You can also select one of three Combo Box Types, as summarized in the following table.

Combo box type	Description
Simple	The drop-down list is always expanded to display items in the list, and the user can edit the items in the list (default).
Drop Down	When the dialog box is first displayed, the combo box consists of a single line of editable text. The user can click the down arrow to expand the list and edit all items in the list.
Drop Down List	Works just like a drop down, but the list is static. The user can select, but can't change, any item in the list.

When sizing combo boxes, remember that the specified size is for the combo box with its list displayed, even if the list is normally hidden.

Scroll bar controls

Both horizontal and vertical scroll bars have only the basic attributes. In addition, you can choose between two types: horizontal and vertical. If you change the orientation of a scroll bar, make sure to also change its boundaries. Although you can have, for example, a tall, narrow horizontal scroll bar, it would not be useful to the user.

Although slider controls act much like scroll bars, they are separate classes of windows with no attributes in common other than the basic attributes.

Slider controls

Slider controls work much like scroll bars, with an arm that the user can slide along a shaft to set a value, with an optional button that moves the arm incrementally in either direction. Like a scroll bar, a slider can be either

of two types: horizontal or vertical. Although the default type is horizontal, you can choose to make any slider vertical.

In addition to the basic attributes, slider controls have the unique styles listed in the following table.

Style	Description
Owner Draw	The dialog box draws the slider control.
Read Only	The user cannot manipulate the control.
Snap To Increment	If the user moves the slider arm to a position between legal values, the control adjusts the position to the nearest legal value.
Ribbon Strip	The control darkens the slider shaft to the left of (or below) the arm.

You can also specify the offset position of the slider shaft within its bounding rectangle, the position of the incremental control button (if any), and the home position of the slider thumb.

Sliders also have two scales associated with them, each specifying the increment and spacing of something. You can choose which of the two scales is the primary (default) scale. Since scales are implemented by the programming API, you can't meaningfully test them within Resource Workshop.

Value set controls

Most of the functions of value set controls are determined by your program, but there are a few properties you can control through the dialog box resource. Because the actual appearance and behavior of the control is dependent on your application, you can't meaningfully test value set controls within Resource Workshop.

The value set style dialog box enables you to set the default type for the items and specify the number of rows and columns in the value set control. The default type can be Bitmap, Text (default), Color Index, Icon, or RGB.

In addition to the basic attributes, value set controls have the additional attributes described in the following table.

Style	Description
Outside Border	Draws a border line around the entire control.
Right To Left	Orders the items from right to left instead of top to bottom.
Item Border	Draws a border around each item in the value set.
Scale Bitmaps	Scales bitmapped images to fit in the item area. By default, only the portion of the full-sized bitmap that fits in the item appears.

Notebook controls

Most of the functions of notebook controls are determined by your program, but there are some basic properties you can control through the dialog box resource. Because the actual appearance and behavior of the control is dependent on your application, you can't meaningfully test notebook controls within Resource Workshop.

The Notebook Style dialog box enables you to specify the basic attributes of the notebook control, the side of the notebook page that has the major tabs, the position of the back page relative to the entire control, the shape of the tabs, and the alignment of the text on tabs and the status line.

Container controls

Most of the functions of container controls are determined by your program, but there are some basic properties you can control through the dialog box resource. Because the actual appearance and behavior of the control is dependent on your application, you can't meaningfully test container controls within Resource Workshop.

In addition to the basic attributes, the Container Style dialog box lets you set the following attributes:

Style	Description
Auto Position	The container automatically rearranges its items when the window changes.
Mini Record	Forces the use of mini record core data structures.
Verify Pointer	Verifies that application pointers are in the container's list before using them.
Read Only	The user cannot modify the container or its contents.

You can also specify the selection type of the container: Single, Extended, or Multiple.

C H A P T E R 5

Using the Bitmap editor

Using the Resource Workshop Bitmap editor, you can create or edit the following bitmapped resources:

- Icons
- Pointers
- Bitmaps

See the README file on the distribution disks for information on where to find sample bitmaps, pointers, and icons.

This chapter describes the following:

- Understanding bitmapped resources
- Using the Bitmap editor
- Using the Bitmap editor tools
- Working with images
- Extra items for icons and pointers

Understanding bitmapped resources

In PM the three bitmapped resource types—bitmaps, pointers, and icons—are virtually identical.

- *Bitmaps* account for a variety of graphic images in your PM program. PM itself uses bitmaps for scroll bar arrows, the Minimize and Maximize buttons, and so on. The Resource Workshop Bitmap editor uses bitmaps for the tools in the Tools palette.
- *Pointers* represent the mouse's current location on the screen. In addition to the familiar arrow pointer, PM uses other pointers to represent different program functions—for example, the Color Palette's paint roller pointer.

  You can create your own special pointers to represent different functions of your application. Note, for example, that each of the Bitmap editor's tools has a pointer that matches the tool's bitmap.

- *Icons* are small bitmapped images that are typically used to represent minimized windows.

Pointers and icons are functionally identical. The only difference between them and bitmaps is that pointers and icons can have transparent and inverted color areas, and pointers and icons can have a *hot spot*. These differences are described later in this chapter.

Bitmapped resource concepts

This section describes three concepts that are basic to using the Bitmap editor:

- Pels
- Left-button and right-button colors
- Images

Pels

Bitmapped resources created with the Bitmap editor are painted on a grid of roughly square "dots" called *pels*. You create the image by using Bitmap editor tools to assign a color to each pel. The pels assemble like a mosaic to form the bitmapped image.

Left-button and right-button colors

The colors you assign to the pels are painted with the left and right buttons of the mouse. For that reason, they are referred to as *left-button* and *right-button* colors.

You can use a variety of selected left-button or right-button colors for the features of your image (lines, boxes, and so on). The ability to assign colors to both mouse buttons means you can have two colors at your disposal at any time.

There is one important difference between the left-button and right-button colors. When you delete or move a block of pels in your image (see page 75), the currently selected right-button color replaces the color in the pels no longer occupied by the block.

To select a color, click the left or right mouse button on the color you want in the Colors palette. If you click with the left button, the letters *LB* appear on that color; if you click with the right button, the letters *RB* appear on the color. If you click the same color with both buttons, the letters *BB* appear on the color.

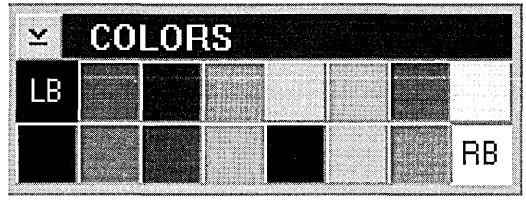


Figure 5.1
Colors palette

The Eraser operates in the opposite fashion from the other tools. The left mouse button produces the color marked *RB*, and the right mouse button produces the color marked *LB*.

Images

Each bitmapped resource must contain a device-independent form of the resource. In addition, the resource can contain multiple *images*, typically the same bitmapped resource in several device-dependent resolutions or color formats.

PM automatically picks an image format that matches the display hardware, if one is available. If one isn't available, PM picks the device-independent image and scales it to the current resolution and color capability.

Depending on the target device, bitmapped resources can be created in resolutions of 32×32, 32×16, or 40×40 pels in size. You can also create custom image resolutions and color combinations.

Using the Bitmap editor

No matter which type of bitmapped resource you're editing, there are certain tasks you'll always perform. This section describes

- Starting the Bitmap editor
- The Bitmap editor screen
- Deleting bitmapped resources

Use of the Bitmap editor's painting tools is in the next section, "Using the Bitmap editor tools," starting on page 74.

Starting the Bitmap editor

Resource Workshop starts the Bitmap editor any time you

- Add a new bitmapped resource to a project
- Create a standalone bitmapped resource file
- Edit an existing bitmapped resource

Adding a bitmapped resource to a project

To add a bitmapped resource to a project,

1. Open a new or existing project. (Chapter 3 describes how you open a project.)
2. Choose Resource | New or double-click POINTER or BITMAP in the project window. Resource Workshop displays the New Resource dialog box.

The New Resource dialog box is described on page 33.

3. Select the bitmapped resource type from the Resource Type list box. This is done automatically if you double-clicked POINTER or BITMAP in the project window.
4. Move to the Resource ID list box and enter a name for the resource.
5. The name of the current project file appears in the entry field under Place Resource In. If you don't want to place the bitmapped resource in the current project file, you can scroll down the list to pick another file (if any is listed), or you can click New to create a new project file.
6. Select an identifier file from the Place Identifiers In list box. To create a new identifier file, click the New button in this panel. Resource Workshop displays the Add File to Project dialog box.
7. Click OK to exit the New Resource dialog box.
8. If you're creating a bitmap, Resource Workshop displays the New Bitmap dialog box. You can either accept the default bitmap size or specify a new size. Click OK to exit the New Bitmap dialog box.
9. Resource Workshop puts the new bitmapped resource name in the Project window and starts the Bitmap editor.

Note that another window appears just before the Bitmap editor itself. This is the *launch window*, which is described on page 79.

Creating a standalone bitmapped resource file

You can create a standalone bitmapped resource file by choosing File | New Project and selecting .ICO, .PTR, or .BMP from the New Project dialog box.

Resource Workshop immediately starts the Bitmap editor for the selected bitmapped resource type.

To link a standalone bitmapped resource file to a multiple-resource project file, open the project file and then choose File | Add to Project. (See page 33.)

Loading an existing bitmapped resource

To load an existing bitmapped resource into the Bitmap editor,

1. Open an existing project. (Chapter 3 describes how to open a project.) Resource Workshop displays the Project window.
2. In the Project window, double-click the name of the bitmapped resource you want to edit, or select it and choose View | Edit Visually.
3. Resource Workshop displays the launch window, which lists all the images in the selected resource. If the resource has only one image, Resource Workshop automatically launches the Bitmap editor.

   In the launch window, find the bitmapped resource image you want to edit. Double-click it, or select it and choose either Icon/Pointer | Edit Visually or Bitmap | Edit Visually, depending on the kind of resource

The launch window is explained starting on page 79.

you're editing. Resource Workshop starts the Bitmap editor with the bitmapped resource loaded.

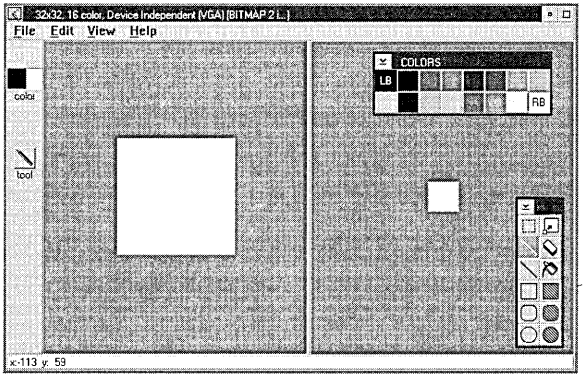
Bitmap editor screen

The Bitmap editor has four main components:

- Window panes
- Colors and Tools palettes
- Sidebar
- Status line

This section describes how to use each one.

Figure 5.2
The bitmap editor screen



The Bitmap editor window has the standard PM window components: title bar, menu bar, System menu, Minimize and Maximize buttons. The title bar contains the text that appears in the launch window—typically the pel resolution of the image, color format, and target device.

Using the window panes

The Bitmap editor window is initially divided vertically into two *panes*. One shows the image in its actual size, and the other shows a close-up view of the image. You can zoom in or out on the image in either pane, divide the window horizontally, and change the relative size of the two panes, including "closing" either pane.

In the Bitmap editor, you can look at two different views of the image you're creating or editing. You can split the window vertically or horizontally to show the two views side-by-side or one view above the other. You can also choose how to zoom each view.

To split the window, choose View | Split Horizontally or View | Split Vertically.

For example, if you split the window vertically, you could display the entire image at its actual size in the right window pane and zoom in on a small portion of the image in the left window pane.

When the window is split, the pane in which you're working is the *active* pane. To make a pane active, click the mouse inside it. The Bitmap editor indicates the active pane by making it appear recessed, like a pushed button or a selected tool.

To see more of one view than the other, move the pointer to the line that splits the images (the separator bar) and, when the pointer becomes a double arrow, drag the separator bar. For example, with the windows split vertically, you can drag the separator bar to the right to see more of a zoomed image.

To remove the split window entirely and return to a single view, drag the separator bar all the way to the left or right (for a vertical split) or to the top or bottom (for a horizontal split). To return to a two-pane view of the image, choose View | Split Horizontally or View | Split Vertically.

Colors palette and Tools palette

Each open Bitmap editor window has its own Colors palette and Tools palette.

- The Colors palette shows the colors you can use in your bitmapped resource and the current button assignments.
- The Tools palette contains the tools with which you'll create your bitmapped resource. You can use these tools to paint lines, filled areas, and outlined shapes, and also to fill areas with color, erase parts of the image, zoom in on the image, or select blocks of pels.

The palettes are windows: you can move, close, and open them.

- To move a palette, drag its title bar.
- To close a palette, double-click its title bar icon or choose Close from its Control menu.
- To open a palette after you've closed it, click the appropriate indicator in the sidebar.

The palettes are not bound by the main Bitmap editor window. If you drag a palette outside the window and then maximize the window, the palette will maintain its position relative to the main window and will be in effect, "off the screen." To bring a palette back into view, click the appropriate indicator in the sidebar, as described in the next section.

Using the sidebar

The Bitmap editor window's sidebar contains a color indicator that shows the current left-button and right-button color and a tool indicator that shows the currently selected tool. You can drag the sidebar to either side of the window.

If you've closed either the Colors palette or the Tools palette, you can reopen them by clicking the sidebar's color indicator or tool indicator, respectively.

Reading the status line

The status line at the bottom of the Bitmap editor window provides information about the pointer position and the editor's current mode. Here is a partial list of the information provided by the status line:

- If the pointer is inside either pane—but not in the image area or on top of a palette—the status line shows the x- and y-coordinates of the pointer's hot spot relative to the lower left corner of the image area.

  Each pane has its own set of coordinates, which are affected by the pane's current zoom level. When you move from one pane to another, you also move from one coordinate system to another.
- If the pointer is inside the image area, the status line shows the pointer's coordinates and information about the color under the pointer.

  If the pointer is over a color from the palette, the status line information includes the color index and its RGB (Red, Green, Blue) values.

  If the pointer is over a Transparent or Inverted color, the status line says "Transparent" or "Inverted."
- If the pointer is over a palette, the status line remains frozen with the last information displayed before the pointer went into the palette.
- If you're setting a hot spot, the status line gives the current hot spot position, the instruction "Click to set new hotspot," and the pointer's position.
- As you click a menu or use accelerator keys to select a menu command, the status line displays more information about the selected command.

  For example, if you display the View menu and press the left mouse button over the Zoom In command, the status line reads `Magnify by a factor of two.`

Deleting bitmapped resources

When you delete a bitmapped resource, you delete all the images in that resource. To delete a bitmapped resource, select it in the Project window and then do one of the following:

- Press the *Del* key or choose Edit | Delete to completely delete it.
- Choose Edit | Cut to cut the resource to the Clipboard so you can paste it elsewhere.

You can also delete individual images within a resource. See page 80.

Using the Bitmap editor tools

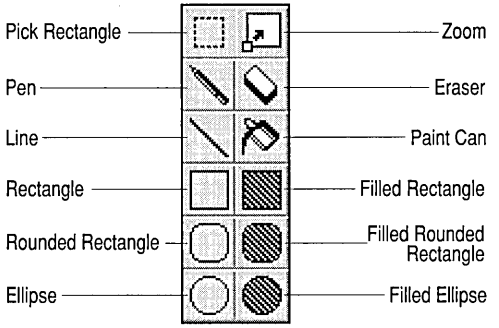
To select a tool from the palette, click the tool you want. The selected tool appears in reverse video. The following sections describe each tool.

The painting tools—Pen tool, Line tool, Paint Can, empty frames, filled frames, and Eraser tool—can use either the current left-button or right-button color.

If you see *BB* in the Colors palette, the same color is selected as the left-button and right-button color.

- Use the left mouse button to paint using the current left-button color (*LB* on the Colors palette).
- Use the right mouse button to paint using the current right-button color (*RB* on the Colors palette).

Figure 5.3 Bitmap editor Tools palette



Working with blocks: Pick Rectangle tool

The Pick Rectangle tool enables you to select a rectangular block of pels by dragging the pointer. This section describes the following block tasks:

- Selecting and deselecting
- Cutting and copying
- Pasting
- Deleting
- Moving
- Duplicating

Selecting and deselecting blocks

To select a block, put the tip of the pointer at one corner of the rectangle and drag to the diagonally opposite corner. When the flashing outline includes the area you want, release the mouse button.

To select the entire image, choose Edit| Select All.

To deselect a block, click the right mouse button inside the Bitmap editor window. You can also deselect a block by clicking the left mouse button anywhere in either pane or by pressing *Enter*.

The mouse buttons and *Enter* key deselect only; if you've moved or copied the block, the pels remain in their new position.

If you press *Esc* after moving or copying a block, and the block is still selected, you both undo the move or copy *and* deselect the block.

You can use the commands in the Edit menu to cut, copy, delete, duplicate, or paste a block, and you can use the mouse to move or duplicate it.

Cutting and copying blocks

The Cut and Copy commands both put the selected block in the Clipboard so you can paste it to another bitmap, either in the same project or in another project running in another copy of Resource Workshop. However, the two commands have a different effect on the current image:

- Cut removes the selected block from the image and replaces it with the currently selected right-button color.
- Copy leaves the selected block in the image.

Pasting blocks

The Paste command copies the contents of the Clipboard into the current window in the following manner:

If the Clipboard image was created with the Bitmap editor, the Paste command places it at the same pel position from which it came.

If, because of differences in image resolutions, the source position would put the Clipboard image outside the target image area, or if the Clipboard image didn't come from the Bitmap editor, the Paste command puts the Clipboard image at the upper left corner of the Bitmap editor image area.

If you select an area in the target and then choose Edit | Paste, the Bitmap editor stretches or shrinks the contents of the Clipboard as necessary to make it fit the selected area.

The selected block remains on the Clipboard and you can continue pasting it until you overwrite it by cutting or copying another block to the Clipboard.

Deleting blocks

The Delete command deletes the selected block and replaces it with the right-button color. Pressing *Del* has the same effect.

Delete resembles Cut, but you can't paste a deleted block. You can only Undo a Delete, restoring the block to its original place.

Moving blocks

To move a block, put the pointer anywhere inside the selection rectangle and drag to a new location. You can also move the block by pressing the arrow keys.

When you select and then move a block, you're in effect pulling it out of the background. Any "open" pels that are left behind are filled with the current right-button color.

You can continue to move the block without further effect on the background. To set the moved block into the image, deselect it by pressing the right mouse button.

Duplicating blocks

Duplicating a block differs from copying in that it doesn't use the Clipboard. For example, you can cut or copy something to the Clipboard, mark another block and duplicate it, and still paste the first block from the Clipboard.

Duplicating resembles copying, however, in that it leaves the original area unchanged.

You can duplicate blocks with the Duplicate command or with the mouse.

- The Duplicate command places a copy of the selected block in the upper left corner of the image frame.
- By holding down the *Ctrl* key as you drag with the mouse, you can duplicate a block to a specific location, not just the upper left corner.

  If you forget to press *Ctrl*, you can still duplicate the block:

  1. Drag back to the initial position if necessary.
  2. Release the mouse button.
  3. Press and hold *Ctrl*.
  4. Perform the drag again.

As long as the duplicated area is still selected, you can use the mouse or arrow keys to move it anywhere in the image. Because the block is temporarily "floating" on top of the image area, you can move it around without affecting the background.

To set the duplicated block into the image, deselect it by clicking the right mouse button outside the block, by clicking the left mouse button anywhere, or by pressing *Enter*.

Zooming images: Zoom tool

You can use the Zoom tool to zoom in or out on the image. This section describes the following zoom tasks:

- Zooming the entire image
- Zooming to a selected area
- Moving zoomed images

Zooming the entire image

The Bitmap editor uses the center of the image as a reference when zooming the entire image.

To zoom in or out on the entire image, you can use any of the following techniques. In each case, Resource Workshop zooms by a factor of two.

- To zoom in, double-click the Zoom tool icon in the Tools palette. To zoom out, press *Shift* as you double-click the Zoom tool icon.
- Choose View | Zoom In or View | Zoom Out.
- Press *Ctrl+Z* to zoom in; press *Ctrl+O* to zoom out.
- To display the image in its actual size, press *Ctrl+A*.

When you're working with two window panes (see page 71), zooming affects only the active window.

When working with a close-up view of the image, use the scroll bars or arrow keys to move the zoomed image around.

Table 5.1 lists all the ways you can zoom.

Table 5.1
Zoom commands

View command	Accelerator key	Mouse action on Zoom icon
Zoom in	*Ctrl+Z*	Double-click
Zoom out	*Ctrl+O*	*Shift*+double-click
Actual size	*Ctrl+A*	None

Zooming to a selected area

To zoom in on a selected area, drag a rectangle in the image with the Zoom tool. When the flashing outline includes the area you want to see, release the mouse button. Resource Workshop zooms the area to the largest zoom percentage that will fit in the window pane (from 100% to 3200%).

Moving zoomed images: Hand tool

You can also use the scroll bars to move the image.

The Bitmap editor's Hand tool is specifically designed for moving images that are zoomed so much that they are partially cut off by the window pane. You can use the Hand tool to bring other parts of the image into view.

The Hand tool isn't included in the Tools palette, but you can temporarily change any tool into a hand by holding down *Ctrl*. (The one exception is the Pick Rectangle tool when there is a selected block and the pointer is inside the block.) Using the hand, you can take hold of the image and drag it inside the frame.

When you release *Ctrl*, the current tool returns.

Painting freehand lines: Pen tool

The Pen tool paints freehand lines one pel wide. To sketch with the Pen tool, press a mouse button and drag the pointer across your image. When you've finished sketching, release the mouse button. (To paint absolutely straight lines, use the Line tool instead of the Pen.)

Erasing and painting: Eraser tool

The Eraser tool can be used to erase the entire image, or it can be used as a painting tool. Note that, for the Eraser, the color assignments to the mouse buttons are the reverse of the other tools' assignments.

- If you double-click the Eraser tool icon in the Tools palette, the entire image is replaced with the current right-button color (*RB* on the Colors palette).
- If you drag with the left mouse button, the Eraser paints a line one pel wide using the current right-button color (*RB* on the Colors palette).
- If you drag with the right mouse button, the Eraser paints a line one pel wide using the current left-button color (*LB* on the Colors palette).

Painting straight lines: Line tool

The Line tool paints straight lines. Press the mouse button and drag the Line tool across your image. When you've finished painting the line, release the mouse button.

To constrain the lines to 45-degree increments (horizontal, vertical, or diagonal), hold down *Shift* as you paint.

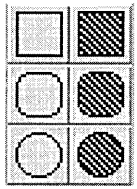
Filling color areas: Paint Can tool

The Paint Can tool floods an area of your image with a color.

To use the Paint Can, place its cross hair in the portion of the image you want to fill and then click the left or right mouse button, depending on the color you want. The Paint Can replaces the color under the pointer with the selected color and fills out around that point until it meets a different color or the image frame.

If you hold down the *Shift* key when you use the Paint Can, you replace *all* instances of the color you click, contiguous or not.

Painting rectangles and ellipses



The bottom three rows of the Tools palette contain tools for painting rectangles, rounded rectangles, and ellipses. The tools in the left column paint an outline only, using the current left-button or right-button color. The tools in the right column paint a solid shape that uses the current left-button or right-button color.

To paint a true square, rounded square, or circle, hold down the *Shift* key as you drag with the mouse.

To use the rectangle, rounded rectangle, or ellipse tool,

1. Select the tool you want.
2. Imagine a rectangular frame enclosing the shape you're going to paint.
3. Place the pointer crosshair at one corner of the imagined frame, and drag to the opposite corner.

   If you're painting an ellipse or circle, note that your starting point is *not* on the shape itself. That's why it's a good idea to picture the rectangular frame first.
4. Release the mouse button when the shape is the way you want.

Working with images: the launch window

In addition to starting the Bitmap editor with the launch window (see page 70), you can use the launch window to add or delete images and create custom images.

Adding an image

As noted previously, each bitmapped resource must include a device-independent image, but you can also create other images, typically the same design at a number of device-dependent resolutions or color schemes.

For example, you might create an icon for EGA and VGA resolutions and also for a very high-resolution device. In that case, you would have four images of the icon (the device-independent image and the three device-dependent images), all saved as part of the one resource. It's also common to create half-size images of icons and pointers for use as the system-menu icon on a window's title bar or when you choose the small icon display in the PM desktop.

To add a new image to an existing bitmapped resource,

1. Bring the launch window to the top, either by clicking it directly or by double-clicking it in the Window List.

2. Choose Icon/Pointer | New Image or Bitmap | New Image or press the *Ins* key. Resource Workshop displays the New Image dialog box.
3. Double-click one of the preset image size and color format combinations, or select an image format and click OK.

Resource Workshop opens a new instance of the Bitmap editor. The new image is a copy (scaled, if necessary) of the most recently saved device-independent image from the launch window. If you've made changes to the device-independent image, but haven't saved them, they will not appear in the new image.

Deleting an image

To remove an image from a bitmapped resource,

1. Bring the launch window to the top, either by clicking it directly or by double-clicking it in the Window List.
2. Select the image you want to delete and then do either of the following:
   - Press the *Del* key or choose Icon/Pointer | Delete Image or Bitmap | Delete Image.
   - Choose Edit | Cut to cut the image to the Clipboard so you can paste it elsewhere.

Creating custom images

You can create custom images with special resolutions and color capabilities. To create a custom image,

1. Bring the launch window to the top, either by clicking it directly or by double-clicking it in the Window List.
2. Choose Icon/Pointer | New Image or Bitmap | New Image. Resource Workshop displays the New Image dialog box.
3. The New Image dialog box contains a list of all the image formats you can edit on the current device that don't already have images. If your custom image is similar to one of these, you can click that one to use as a starting point before moving to the next step.
4. Click the Edit button to display the Edit Image Type dialog box.
5. Fill out the Edit Image Type dialog box, as described in the next sections, clicking OK when you finish.

When you finish, if the contents of the Edit Image Type dialog box exactly match an existing image type, Resource Workshop won't create a new type. At least one of the fields described in the following sections must differ from existing types.

Image Type description

This field contains the text description of the image type. It appears in the title bar of the Bitmap editor window for that image, in the launch window, and in the Window List.

You can enter a new description in this field, or you can select a description from the list box and edit it.

The Image Type Description for each image of a given bitmapped resource must be unique.

Nominal Image Size

The Nominal Image Size is the width and height of the image in pels. For example, for VGA screens the default Nominal Image Size for bitmaps, icons, and pointers is 32×32 pels; for the 8514 display device it is 40×40 pels.

You should specify a Nominal Image Size that is proportionally appropriate to the target device. The Info button in the New Image dialog box provides this information for the current device.

Device Size

The Device Size is the actual pel size of the target device—for example, 640×480 for a VGA or 1024×768 for an 8514, or 0×0 for a device-independent image.

Device Resolution

The Device Resolution is the number of vertical and horizontal pels *per meter* on the target device.

If you're not sure of this value, you can set it to zero.

Extra items for icons and pointers

Icons and pointers in PM are functionally identical. This section describes three tasks that are unique to icons and pointers.

- Using transparent and inverted colors
- Setting the pointer's hot spot
- Testing icons and pointers

Using transparent and inverted colors

The idea of a *transparent* or *inverted* color is unique to icon and pointer resources.

- A transparent color "drops out" at run time, allowing whatever is underneath the icon or pointer to show through. This is especially useful

in pointers, where you will typically not use the entire image area for the pointer itself.

- Inverted colors cause whatever is underneath the icon or pointer to "reverse" at run time.

For example, you could create an icon made of a black outline around two rectangles, one of the transparent color and the other of the inverted color. If you test the icon by dragging it over the color black in the Colors palette, you'll see black in the transparent color part of the icon and white (the inverse of black) in the inverted color part. If you drag the icon over the blue in the Colors palette, you'll see blue and yellow, respectively.

The transparent color is the current desktop color set in the OS/2 System Setup's color palette.

To use the transparent or inverted color, click it with the mouse to designate it the left-button or right-button color.

Setting the pointer's hot spot

Whatever its design, a pointer always "touches" the application's work surface with a single pel. For example, if you look at some of the Bitmap editor's tool pointers, you'll see a paint can, pen, or filled rectangle, each with a crosshair of intersecting horizontal and vertical lines. The pointer's hot spot is the single pel where the two lines intersect.

To set the hot spot, do the following:

1. If necessary, zoom in on the image so you can set the hot spot accurately.
2. Choose Image | Set Hot Spot. The pointer becomes a crosshair in a circle.
3. Place the crosshair where you want the hot spot, and click.

After you set the hot spot, you can test it by choosing Image | Test. If you set the hot spot at the top edge of the pointer, the test pointer will change to the Pick Rectangle tool pointer as soon as you touch the menu bar. If you set the hot spot at the bottom edge of the pointer, you will be able to move almost the entire test pointer up into the menu bar. To end the pointer test, click anywhere inside the Bitmap editor's workspace (not in the menu bar, palettes, sidebar, or scroll bars).

Icons and pointers always have a hot spot. The default position for the hot spot, if you don't explicitly set one, is the center of the image.

Testing icons and pointers

To test your icon or pointer, choose Image | Test. The Bitmap editor pointer changes to the icon or pointer you've designed. Among other things, you can test for the following:

- The icon or pointer's general appearance
- How the Transparent and Inverted areas look over the Bitmap editor window
- The behavior of the icon or pointer when it moves into the window's sidebar, menu bar, or border
- The behavior of the hot spot (if any)

To end the icon/pointer test, click anywhere inside the Bitmap editor's workspace (not in the menu bar, palettes, sidebar, or scroll bars).

C H A P T E R 6

Using the Script editor

Other than bitmapped images and dialog boxes, OS/2 PM resources are created and changed by editing *resource scripts*. A resource script is a text file that defines one or more resources.

This chapter describes

- Using the resource script editor
- The syntax of a resource script
- Using each kind of resource script statement

Resource Workshop always edits some kinds of resources as scripts. These resources are referred to as *scripted resources*:

- Accelerators
- Association tables
- Fonts
- Help tables and subtables
- Menus
- Message tables
- String tables
- User-defined (custom) resources

You can also define bitmapped and dialog template resources with scripts, but you'll normally find it easier to use the visual editors in Resource Workshop, especially for creating those resources. You'll probably edit bitmap or dialog template scripts only to make changes to existing resources.

Using the script editor

This section describes when and how to use the script editor.

When to use the script editor

You can use the script editor to edit any resource by selecting the resource in the project window and choosing the Resource | Edit As Text menu command. When you choose a scripted resource from the project window or add a scripted resource to the project, Resource Workshop always starts the script editor for you.

How to use the script editor

The script editor is a standard text editor, with the same kind of editing capabilities found in the OS/2 system editor. In addition to the ability to edit resource files, you can compile the resource being edited by choosing the Resource | Compile menu command.

Each script editor handles only a single resource. When you choose a particular resource, Resource Workshop loads just the script for that one resource into the script editor, even if that resource is part of a larger script that contains many resources. When you close the script editor, Resource Workshop updates the larger script with your changes.

For more details on using the Script editor, see "Using the internal Script editor" on page 32.

The default resource template

When you start the script editor for a new resource, the editor contains a blank template for the resource, consisting of a resource keyword, a resource ID, and an empty BEGIN and END. For example, the template for a new message table with an ID of 42 would look like this:

```
MESSAGETABLE 42
BEGIN
END
```

Closing the script editor

When you finish editing a resource script, close the script editor window. Resource Workshop automatically updates the project.

Writing resource scripts

A resource script is a text file that consists of three kinds of items:

- Resource definition statements
- Resource compiler directives
- Comments

Resource definition statements

The majority of a resource script is made up of resource definition statements. Each resource in the resource script is defined by a definition statement, which has the following general syntax:

```
keyword resource_id [options]
BEGIN
items
   :
END
```

In this syntax, `keyword` is one of the resource key words listed in Table 6.1, `resource_id` is either a numeric constant or an identifier, and `options` is optional information specific to the type of resource being defined. `items` is one or more items within the resource, such as menu items in a menu resource or strings in a string table.

Table 6.1
Recognized resource definition keywords

Keyword	Resource type	Page
ACCELTABLE	Accelerator table	88
ASSOCTABLE	File-association table	91
FONT	Font	93
HELPITEM	Help item	94
HELPSUBITEM	Help subitem	96
HELPSUBTABLE	Help subtable	95
HELPTABLE	Help table	93
MENU	Menu	97
MENUITEM	Menu item	100
MESSAGETABLE	Message table	102
PRESPARAMS	Presentation parameters	103
RCDATA	Custom resource	90
STRINGTABLE	String table	104
SUBMENU	Submenu	105

For examples of each keyword, see the alphabetical lookup section at the end of this chapter.

Resource compiler directives

Your resource script can include compiler directives, much like C++ preprocessor directives. The most commonly used directives are **#include** and **#define**, which work just like their C++ counterparts, and **rcinclude**, which includes another resource script. Table 6.2 lists all the supported directives.

Table 6.2
Supported resource compiler directives

#define	#endif	#ifdef	#line
#elif	#error	#ifndef	#pragma
#else	#if	#include	#undef

Comments

Any line in the file that begins with an asterisk (*) is ignored by Resource Workshop. You can insert comments anywhere in a script file.

Resource script reference

For additional information on the scripted resources, including specific PM function and message names, see the Resource Workshop Help.

The description of each resource provides the following information:

- A description of the resource type
- Its resource script syntax
- Descriptions of the script options or parameters
- A brief example of the resource script

In the syntax descriptions in the following sections, optional items are enclosed in square brackets []. Items in uppercase letters (like `ACCELTABLE`) must be entered exactly as given, although you can use lowercase letters. Items in lowercase letters (like `acctbl_id`) are variables for which you substitute a value.

Several of the scripted resources have load options (`load_opts`) and memory options (`mem_opts`) in their syntax. The load and memory options are described in Table 3.2 on page 37. These options are themselves optional. If you don't explicitly specify an option, Resource Workshop automatically assigns the default value, which is identified in the sections on the individual resources.

Each resource must have a resource ID, which must be unique *within each resource type*. For example, an ACCELTABLE resource and a MENU resource could both have ID values of 1, but you can't create two ACCELTABLE resources with an ID value of 1.

Accelerator tables — ACCELTABLE

The ACCELTABLE resource is a table of accelerators for your application. Accelerators are keyboard combinations that serve as shortcuts for menu commands, so the user doesn't have to display the menu and click the command name. You can also create accelerator resources that define new functions not available from your program's menus.

By default, an accelerator sends a WM_COMMAND message to the application. Exceptions to this default are noted in Table 6.3.

Syntax

```
ACCELTABLE acctbl_id [mem_opts]
BEGIN
key_val, command[, acc_opts]
   ⋮
END
```

The default memory option for the ACCELTABLE resource is MOVEABLE.

acctbl_id The accelerator table identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple ACCELTABLE resources, each of which must have a unique acctbl_id value.

key_val The character, scan, or virtual-key code for the accelerator key. You must enter a single character in double quotation marks or a decimal or hexadecimal integer ranging from 0 through 255. The entry in this field is tied to the entry in the acc_opts field.

By default, the accelerator is assumed to be of type CHAR. If you enter an integer, you must explicitly state the CHAR, SCANCODE, or VIRTUALKEY options.

command The identifier of the command to which the accelerator is linked. This value must be an integer ranging from 0 through 65535 or a simple expression that produces a value in that range.

acc_opts This field defines the accelerator's type. You can combine most of the values listed in Table 6.3. VIRTUALKEY, SCANCODE, and CHAR are mutually exclusive, however, as are SYSCOMMAND and HELP.

Table 6.3 Accelerator option values

Option	Description
VIRTUALKEY	Identifies key_val as a virtual key code—a function key, for example.
SCANCODE	Identifies key_val as a keyboard scan code.
CHAR	Identifies key_val as a character code.
SHIFT	The accelerator combines key_val with the *Shift* key.
CONTROL	The accelerator combines key_val with the *Ctrl* key.
ALT	The accelerator combines key_val with the *Alt* key.
LONEKEY	The accelerator uses only the key identified by key_val.
SYSCOMMAND	The accelerator causes the application to send a WM_SYSCOMMAND message.
HELP	The accelerator causes the application to send a WM_HELP message.

Example

```
ACCELTABLE 1
BEGIN
"O", menu1_Open, CONTROL
"S", menu1_Save, CONTROL
"P", 101, ALT,
"Q", 102, CONTROL, ALT
"H", menu1_Help, ALT, HELP
END
```

This accelerator table has the following characteristics:

- It has an ID value of 1 and uses the default memory option, MOVEABLE.
- The first two accelerators in the table—*Ctrl+O* (Open command) and *Ctrl+S* (Save command)—are tied to their respective commands by the identifiers menu1_Open and menu1_Save.
- The next two accelerators—*Alt+P* (Print command) and *Ctrl+Alt+Q* (Quit command)—are tied to their commands by command ID numbers.
- The first four accelerators send a WM_COMMAND message to the application.
- The last accelerator—*Alt+H* (Help command)—uses the identifier menu1_Help and sends a WM_HELP message to the application.

Custom resources — RCDATA

The RCDATA resource defines an application's custom resources, whose format is entirely up to the application.

Syntax

```
RCDATA rcdata_id
BEGIN
rcdata_def, rcdata_def, ...
   ⋮
END
```

rcdata_id

The RCDATA resource identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple RCDATA resources, each of which must have a unique rcdata_id value.

rcdata_def The application-specific custom resource data, which can be a simple expression or a string.

Example

```
RCDATA 12
BEGIN
    "Veronica Manganese", "Victoria Wren", Botticelli
END
```

This RCDATA resource has an ID value of 12. What happens to the data in this resource is entirely up to the application.

File-association tables — ASSOCTABLE

The ASSOCTABLE resource creates a *file-association table*, which links data files to applications that can edit them. When the user double-clicks a data file or performs a drag-and-drop, PM automatically starts the associated application and loads the selected file.

A file-association table can also associate icons with data files. All data files of a given type must have the same unique icon.

The file associations are attached to the application's executable file. To view an application's file associations, look at the Association page of the executable file's Settings notebook.

Syntax

```
ASSOCTABLE assoctbl_id [load_opts] [mem_opts]
BEGIN
assoc_name, file_match_string[, ext_attr_flag][, icon_filename]
   ⋮
END
```

The default load option for the ASSOCTABLE resource is LOADONCALL; the default memory option is MOVEABLE.

assoctbl_id The association table identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple ASSOCTABLE resources, each of which must have a unique assoctbl_id value.

assoc_name The name of the file type the application recognizes. Valid characters must be in the range from 1 through 255 and must be enclosed in double quotation marks. To use quotation marks as a literal value in this field, use quotation marks twice ("").

file_match_string The file-matching string of the type of data file the application creates. Entries in this field must be enclosed in double quotation marks and can use only valid OS/2 file name and extension characters, plus the question mark (?) and asterisk (*) wildcard characters.

ext_attr_flag The following table describes the ASSOCTABLE resource's extended-attribute options, which can be used in combination.

Table 6.4
File-association table extended attribute flag values

Option	Description
EAF_DEFAULTOWNER	The application containing the file-association table starts when the user selects any file matching the file_match_string field from the File Manager.
EAF_REUSEICON	The icon defined in the previous entry of the file-association table is used as the icon for the current data file type.
EAF_UNCHANGEABLE	The entry cannot be edited.

icon_filename The name of the file containing the icon that represents all data files that match the file_match_string field. The icon file must be in the current directory.

Example

```
ASSOCTABLE 1492
BEGIN
    "Borland Super Spreadsheet", "*.bss", EAF_DEFAULTOWNER, bss.ico
    "Leafy Green Vegetable", "*.lgv", EAF_DEFAULTOWNER | EAF_REUSEICON
END
```

This example file-association table has an ID of 1492, and associates two kinds of files with the application.

The first kind of file has either the file type "Borland Super Spreadsheet" or an extension of .BSS. Files of that type have the icon defined in the file BSS.ICO.

The second kind of file has either the file type "Leafy Green Vegetable" or an extension of .LGV. Files of this type use the same icon already defined, in this case, BSS.ICO.

Choosing either of those kinds of files starts the application.

Fonts FONT

The FONT resource references a font file containing one or more bitmapped "characters." The characters in the font file are typically created with the OS/2 Presentation Manager Font Editor. They can be text characters or images that are used in the application's user interface—for example, the bitmaps that appear in the application's tool bar.

Syntax

```
FONT font_id [load_opts] [mem_opts] filename
```

The default load option for the FONT resource is LOADONCALL; the default memory options are MOVEABLE and DISCARDABLE.

font_id The font resource identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. Each font_id value in the script file must be unique.

filename The name of the file containing the font resource. If the file is not in the current directory, you must provide a full path.

Example

```
FONT 5 toolbar.fon
```

This FONT resource has a font_id value of 5, and the resource data is contained in the file TOOLBAR.FON.

Help tables HELPTABLE

The HELPTABLE resource defines a help table, whose entries let the application access requested help data for application windows, dialog boxes, and message boxes.

An application can have multiple associated help files. The application generates one or more help tables from its help files. The entries in the help table point to further entries in a help subtable, which in turn point to the actual help text.

Syntax

```
HELPTABLE helptbl_id
BEGIN
helpitem_def
   ⋮
END
```

helptbl_id The help table identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple HELPTABLE resources, each of which must have a unique helptbl_id value.

helpitem_def The HELPITEM definition. See the definition of help items starting on page 94.

Example

```
HELPTABLE 1
BEGIN
    HELPITEM IDWIN_FILEMENU, IDSUB_FILEMENU, IDEXT_APPHLP
    HELPITEM IDWIN_EDITMENU, IDSUB_EDITMENU, IDEXT_APPHLP
END
```

This help table's resource ID is 1, and it contains two help items.

The first HELPITEM statement contains the identifier (IDWIN_FILEMENU) of an application window for which help is available. The second entry is the identifier (IDSUB_FILEMENU) of the help subtable for that application window. The last entry is the identifier (IDEXT_APPHLP) of the extended help panel for that application window.

The second HELPITEM statement contains identifiers for another application window and help subtable. Note, however, that both help items share the same extended help panel.

Help items — HELPITEM

The HELPITEM statement, which is permitted only in a HELPTABLE resource, defines the help items in a help table.

A help table can contain multiple HELPITEM statements. There should be a HELPITEM statement for each application window with associated help data.

Syntax

```
HELPITEM app_win_id, help_subtbl_id, ext_hpanel_id
```

app_win_id The resource ID of an application window with associated help data.

help_subtbl_id The resource ID of the help subtable for the application window identified by app_win_id. Help subtables are described starting on page 96.

ext_hpanel_id The resource ID of the extended help panel for the application window identified by app_win_id.

Example

For examples of help items, see the example under Help Tables on page 94.

Help subtables — HELPSUBTABLE

The HELPSUBTABLE resource defines a help subtable, which contains entries for each item in the application window (control, child window, or menu item) for which help is available.

Each item in the help subtable is a child window of the application window named in a HELPITEM statement in the HELPTABLE resource.

Syntax

```
HELPSUBTABLE helpsubtbl_id
[SUBITEMSIZE size]
BEGIN
helpsubitem_def
   ⋮
END
```

helpsubtbl_id The help subtable identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple HELPSUBTABLE resources, each of which must have a unique helpsubtbl_id value.

SUBITEMSIZE As noted in the section on the HELPSUBITEM statement, the default (and minimum) size for help subitems is two words. If you don't use the default, you must specify the size in words of your help subitems.

helpsubitem_def The HELPSUBITEM definition. Help subitems are described beginning on page 96.

Example

```
HELPSUBTABLE IDSUB_FILEMENU
BEGIN
    HELPSUBITEM IDCLD_OPEN, IDPNL_OPEN
    HELPSUBITEM IDCLD_SAVE, IDPNL_SAVE
END

HELPSUBTABLE IDSUB_EDITMENU
SUBITEMSIZE 3
BEGIN
    HELPSUBITEM IDCLD_CUT, IDPNL_CUT, 5
    HELPSUBITEM IDCLD_COPY, IDPNL_COPY, 6
END
```

This example illustrates what the two help subtables in the HELPTABLE example might look like.

The first help subtable has the ID value IDSUB_FILEMENU, which ties it to the first HELPITEM in the help table. The parent window is the File menu, and the child windows are the Open and Save commands.

The second help subtable has the ID value IDSUB_EDITMENU, which ties it to the second HELPITEM in the help table. The parent window is the Edit menu, and the child windows are the Cut and Copy commands. This help subtable doesn't use the default help subitem size, so the size of 3 words is explicitly stated with the SUBITEMSIZE statement. The meaning of the third word in each subitem (5 and 6 in this case) is defined by your application.

Help subitems — HELPSUBITEM

The HELPSUBITEM statement, which is permitted only in a HELPSUBTABLE resource, defines the help subitems in a help subtable.

A help subtable can contain multiple HELPSUBITEM statements. There should be a HELPSUBITEM statement for each child window with associated help data.

Syntax

```
HELPSUBITEM child_win_id, helppanel_id [, integer]
```

chil_win_id The resource ID of a child window with associated help data.

helppanel_id The resource ID of the help panel for the child window identified by child_win_id.

integer Optional, application-defined integers. If you use this value, you must also use the SUBITEMSIZE statement to give the size, in words, of the help subitems in this HELPSUBTABLE resource.

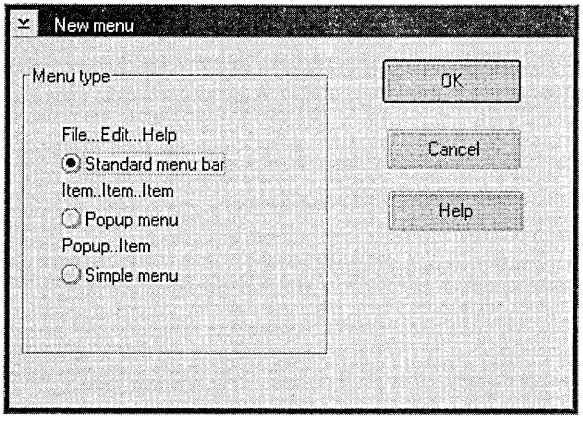
Example

For examples of help subitems, see the example under Help Subtables on page 96.

Menus MENU

The MENU resource defines the contents of a menu, which lists the application's commands. Choosing the MENU resource differs from the other scripted resources because it doesn't immediately start the text editor. Instead, Resource Workshop displays the New Menu dialog box.

Figure 6.1
New Menu dialog box



As Figure 6.1 shows, the New Menu dialog box contains radio buttons for three options:

- Standard Menu Bar
- Popup Menu
- Simple Menu

Each option starts the text editor with a different menu resource script template loaded.

- The Standard Menu Bar template contains a complete script for a menu bar with standard File, Edit, and Help menus.
- The Popup Menu option contains the template for the script of a *popup* menu, one that isn't tied to a menu bar. Popup menus can be displayed inside a window, on the desktop, or from the title-bar icon of a window. To display a popup menu, the user clicks the right mouse button.

  The script for each popup menu must be saved as a separate resource in your project file.
- The Simple Menu option contains the template for the script of a menu that is tied to the menu bar. It contains generic text strings for the menu bar title and a single menu command. You can use this template as the starting point for a menu resource or to paste into an existing script.

The remainder of this section describes the components of the MENU resource.

Syntax

```
MENU menu_id [load_opts] [mem_opts]
BEGIN
menuitem_def
   ⋮
END
```

The default load option for a MENU resource is LOADONCALL; the default memory options are MOVEABLE and DISCARDABLE.

menu_id The menu identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple MENU resources, each of which must have a unique menu_id value.

menuitem_def The definition of the individual menu items, which can be SUBMENU, MENUITEM, or PRESPARAMS statements. These statements are described in the next three sections.

Example

```
MENU main_menu
BEGIN
    SUBMENU "~File", appmen_File
    BEGIN
          MENUITEM "~New\tCtrl+N", appmen_File_New
          MENUITEM "~Open\tCtrl+O", appmen_File_Open
          MENUITEM SEPARATOR
          MENUITEM "~Save\tCtrl+S", appmen_File_Save
          MENUITEM "Save ~as\tCtrl+A", appmen_File_Saveas
          MENUITEM SEPARATOR
          SUBMENU "Set ~destination", appmen_File_Setdest
          BEGIN
              MENUITEM "Set the destination", MIS_STATIC
              MENUITEM "for all file activities", MIS_STATIC
              MENUITEM SEPARATOR
              MENUITEM "~Local\tAlt+L", setdest_local, MIA_CHECKED
              MENUITEM "~Network\tAlt+N", setdest_net
          END
          MENUITEM SEPARATOR
          MENUITEM "~Help\tF1", MIS_HELP
    END

    SUBMENU "~Edit", appmen_Edit
    BEGIN
          MENUITEM "Cu~t\tShift+Del", appmen_Edit_Cut
          MENUITEM "~Copt\tCtrl+Ins", appmen_Edit_Copy
          MENUITEM "~Paste\tShift+Ins", appmen_Edit_Paste
    END
END
```

This MENU resource script defines a main menu bar with two submenus, File and Edit. Note the following about this MENU resource script:

- The identifier names for the MENU resource, SUBMENU statements, and the individual MENUITEM statements.
- The tilde in each menu item (including the SUBMENU statements) marking the mnemonic characters.
- The character (\t) that tabs the accelerators to the right of the command text.
- The MENUITEM SEPARATOR statements that separate the File menu into logical divisions.
- The first two lines of the Set Destination submenu. Because they're instructional and shouldn't be chosen, they have the MIS_STATIC style.

The Local menu item has the attribute MIA_CHECKED. It will be checked when the submenu is first displayed.

- The Help command menu item. It has the style MIS_HELP, which causes it to send a WM_HELP message. All other menu items send a WM_COMMAND message.

Menu items — MENUITEM

The MENUITEM statement contains the command names and accelerators (if any) that appear in menus, plus the menu item's identifier. The MENUITEM statement can appear only within a MENU or SUBMENU statement.

Unless otherwise specified (see the descriptions of the menuitem_style options), a MENUITEM statement sends a WM_COMMAND message to the menu's owner window.

Syntax

```
MENUITEM text, menuitem_id, menuitem_style, menuitem_attr
```

text

The MENUITEM text, which is typically the command text, must be enclosed in double quotation marks. To use quotation marks as a literal character, enter them twice ("").

MENUITEM text uses the following special characters:

- A \t character inserts a tab, forcing all trailing text to the right. All tabbed text is left-aligned. The \t character is commonly used to separate the command text from its accelerator.
- A \a character right-aligns all trailing text.
- A tilde (~) causes the character immediately following to be underlined, indicating that it is a mnemonic for the command in which it appears. To display a menu, the user presses the *Alt* key in combination with the mnemonic key. To choose a command, the user presses the mnemonic key alone.

If the menuitem_style value is MIS_BITMAP (see Table 6.5), the text field must contain a bitmap identifier that has been defined previously with a BITMAP statement. The identifier must be preceded by a \b character and must be enclosed in double quotation marks.

menuitem_id

The menu item identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can

contain multiple MENUITEM statements, each of which must have a unique menuitem_id value.

menuitem_style

These options specify the style of the menu item. By default, a menu item is of type MIS_TEXT.

Table 6.5 menuitem_style options

Option	Description
MIS_BITMAP	The text field contains a bitmap identifier. See the description of the text field.
MIS_BREAK	Causes a pull-down menu to have multiple columns of items or a top-level menu to have multiple lines of menus.
MIS_BREAKSEPARATOR	Draws a vertical line between the columns of a pull-down menu.
MIS_BUTTONSEPARATOR	The user can choose menu items only with the mouse. Menu item text is centered.
MIS_HELP	The menu item sends a WM_HELP message.
MIS_OWNERDRAW	The menu item is drawn by the owner window.
MIS_SEPARATOR	The menu item is a menu separator. You must provide something for the text and menuitem_id fields, although the system ignores it. Instead of using this style, you can use the MENUITEM SEPARATOR statement, which is described after this table.
MIS_STATIC	The menu item is static text. The user can't choose it.
MIS_SUBMENU	The menu item is treated as if it were a SUBMENU statement. The syntax for this item is the same as for a SUBMENU statement, including BEGIN and END clauses and optional PRESPARAMS after the BEGIN clause.
MIS_SYSCOMMAND	The menu item sends a WM_SYSCOMMAND message.
MIS_TEXT	The text field contains a character string.

Instead of using the MIS_SEPARATOR style, you can use the MENUITEM SEPARATOR statement. This statement inserts a horizontal dividing line into the menu. The MENUITEM SEPARATOR statement has no associated text or menuitem_id value.

menuitem_attr

You can assign any combination of the following attributes to your menu items.

Table 6.6 menuitem_attr options

Option	Description
MIA_CHECKED	A check mark appears next to the menu item text.
MIA_DISABLED	The menu item is initially disabled. The user can't choose the item until the application enables it.
MIA_FRAMED	Draws a frame (a heavy border) around the menu item.
MIA_HILITED	The menu item is highlighted (displayed in reverse colors) when first displayed.
MIA_NODISMISS	The menu item or submenu continues to be displayed after the user has chosen an item.

Example

For examples of menu items, see the example under Menus on page 99.

Message tables — MESSAGETABLE

The MESSAGETABLE resource provides one means of storing string data in your application. (The other, the string table, is more commonly used.) String data can appear virtually anywhere in an application, but it is most frequently used in status lines and error messages.

The message table data can be loaded with the DosGetResource or DosGetResource2 function with the RT_MESSAGE resource type.

Syntax

```
MESSAGETABLE [load_opts] [mem_opts]
BEGIN
string_id string_def
   ⋮
END
```

The default load option for the MESSAGETABLE resource is LOADONCALL; the default memory options are MOVEABLE and DISCARDABLE.

string_id

The string identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The string_id can be given in decimal or hexadecimal form, and each value in the script file must be unique.

string_def

The string_def contains the character string and must be enclosed in double quotation marks. To use quotation marks as a literal character, enter them twice (""). You can make character strings continue over additional lines in either of these ways:

- By terminating each line, except the last, with a backslash (\).
- By terminating each line with double quotes and then beginning the next line with double quotes.

Example

```
MESSAGETABLE
BEGIN
    1 "Watch this space"
    2 "You can lead a horse to water,\
       but you can't make him dance."
    3 "If something doesn't happen soon"
      "re-boot your system."
END
```

Note the string_id values preceding the string text. The second and third strings illustrate the two ways of creating multiline messages.

Presentation parameters — PRESPARAMS

The PRESPARAMS statement inserts *presentation parameters* that determine how the menu item (or items) appear when displayed. The PRESPARAMS statement should be entered immediately following the BEGIN statement and remains in effect until it encounters the matching END statement.

Syntax

```
PRESPARAMS presparam, value, presparam, value, ...
```

The presparam field contains a presentation-field type, and the value field contains a presentation-field value.

In the following example,

```
BEGIN
    PRESPARAMS PP_FONTNAMESIZE, "10.Helv"
    MENUITEM "Go", 100
END
```

the PRESPARAMS statement causes the menu item Go to be displayed in 10-point Helvetica type.

For a complete description of the PRESPARAMS statement, see the Resource Workshop Help.

String tables — STRINGTABLE

The STRINGTABLE resource is, compared to the MESSAGETABLE resource, the more commonly used means of storing string data in your application. String data can appear virtually anywhere in an application, but it is perhaps most frequently used in status lines and error messages.

The string table data can be loaded with the WinLoadString function.

Syntax

```
STRINGTABLE [load_opts] [mem_opts]
BEGIN
string_id string_def
   ⋮
END
```

The default load option for the STRINGTABLE resource is LOADONCALL; the default memory options are MOVEABLE and DISCARDABLE.

string_id

The string identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The string_id can be given in decimal or hexadecimal form, and each value in the script file must be unique.

string_def

The string_def contains the character string and must be enclosed in double quotation marks. To use quotation marks as a literal character, enter them twice (""). You can make character strings continue over additional lines in either of these ways:

- By terminating each line, except the last, with a backslash (\).
- By terminating each line with double quotes and then beginning the next line with double quotes.

Example

```
#define str_watch 1
#define str_horse 2
```

```
#define str_crash 3

STRINGTABLE
BEGIN
    str_watch "Watch this space"
    str_horse "You can lead a horse to water,\
               but you can't make him dance."
    str_crash "If something doesn't happen soon"
              "re-boot your system."
END
```

The **#define**s for this STRINGTABLE resource are kept in the project's identifier file.

Note the string identifier names preceding the string text. The second and third strings illustrate the two ways of creating multiline messages.

Submenus — SUBMENU

The SUBMENU statement causes a menu name to appear in the menu bar (a *top-level* menu) or a menu to be displayed from within a menu (a *cascaded* menu).

Syntax

```
SUBMENU text, submenu_id, menuitem_style
BEGIN
menuitem_def
   ⋮
END
```

For a description of the menu items within a submenu (menuitem_def), see the entry for menu items, starting on page 94.

text The SUBMENU text, which is the name of the top-level or cascaded menu, must be enclosed in double quotation marks. To use quotation marks as a literal character, enter them twice (""). You can also use a tilde to indicate the mnemonic character. (See page 100.)

submenu_id The submenu identifier, an integer ranging from 0 through 65535 or a simple expression that produces a value in that range. The script file can contain multiple SUBMENU statements, each of which must have a unique submenu_id value.

menuitem_styl The menuitem_style options are described in Table 6.5.

A P P E N D I X A

Technical notes

This appendix contains technical information on the differences between Resource Workshop and the IBM Resource Compiler.

Compiler differences

The Resource Workshop resource compiler is almost completely IBM-compatible and is significantly enhanced over the IBM Resource Compiler in a number of ways.

The following features are improvements over the IBM compiler:

- The Resource Workshop compiler allows text descriptions of bitmapped resources (icons, pointers, bitmaps, and fonts), while the IBM compiler does not. The text descriptions are written using the resource script language documented in Resource Workshop's online help system.
- The Resource Workshop compiler supports numeric constant expressions for every numeric field, while the IBM compiler doesn't. For example, the following statement is correctly interpreted in Resource Workshop, but causes an error with the IBM compiler:

  ```
  POINTER 20 + 20 - 20 foo.ptr
  ```

  Resource Workshop interprets this line as defining a POINTER resource with an ID of 20 contained in the file FOO.PTR. The IBM compiler would attempt to parse this line as a resource type POINTER with ID 20 contained in the file +.
- Resource Workshop has added a new fundamental data type, the *hexstring*. This data type consists of a variable number of hexadecimal digits that describe data bytes, surrounded by single quotation marks. You can also insert spaces for clarity; the compiler ignores them. This new type makes it easier for users to enter hex data. For example, the following hexstring represents a 5-byte hexadecimal number: `'010A0B0c0E'`. You could also represent this number as follows: `'01 0A 0B 0c 0E'`.

- The Resource Workshop compiler supports references to files in RCDATA resources as well as in user-defined resources. Support of file references removes the only distinction between user-defined resources and RCDATA resources. If you use the IBM resource compiler to compile an RCDATA resource that contains a file reference, you'll get a syntax error.

Note, however, that the IBM Resource Compiler and Resource Workshop are incompatible in the following areas:

- Interpretation of numbers with leading zeros
- The **#undef** preprocessor directive
- Preprocessor token pasting
- Expressions in resources IDs and resource type IDs
- Complex constant expressions
- Floating operators in expressions
- Missing operators in expressions
- Macros in include directives

Numbers with leading zeros

Because of inconsistencies in the IBM Resource Compiler's treatment of numbers with leading zeros, don't use them in preprocessor expressions or identifiers. The Resource Workshop compiler is consistent in interpreting any numeric constant preceded by a zero and used as part of an identifier or a preprocessor expression as an octal number. However, the IBM Resource Compiler interprets numbers with leading zeros in preprocessor expressions as octal numbers, but interprets the same numbers in identifiers as decimal numbers.

For example, the IBM Resource Compiler would interpret the expression 010+1 as a 9 in the following preprocessor expression, but as an 11 in the string table identifier.

```
#if (9 == 010+1)
   STRINGTABLE
   BEGIN
       010+1, "Bug"
   END
#endif
```

#undef preprocessor directive

Resource Workshop has limited support for the **#undef** preprocessor directive. You can use it only with **#define**s that are not referenced by a resource. If you use **#undef** with a **#define** that's a resource identifier, you get a fatal compiler error when compiling the RC file under Resource Workshop.

Token pasting

Resource Workshop does not support token pasting in preprocessor statements. See the *Borland C++ Programmer's Guide* for more information on token pasting.

Expressions in resource IDs and resource type IDs

Resource Workshop supports expressions in resource IDs; the IBM Resource Compiler does not. For example, the following statement compiles correctly using Resource Workshop, but fails using the IBM Resource Compiler:

```
BITMAP 101 + 1000 vga.bmp
```

The IBM Resource Compiler parses "101" as a resource ID, "+" as a file name, and then fails. Resource Workshop correctly emits a bitmap resource with an ID equal to 1101.

Complex constant expressions

Resource Workshop supports full C-language constant expressions in place of a simple number anywhere in a resource script where a number is allowed. The IBM Resource Compiler supports only simple expressions. For example, the following expression is correctly evaluated by Resource Workshop, but fails using the IBM Resource Compiler:

```
3 * (1 + 2) - 1
```

The most common example of this incompatibility is often seen in ICON statements in DIALOG templates. The following statement causes an error in Resource Workshop, but not in the IBM Resource Compiler:

```
ICON   3 -1, 10, 10, 0, 0
```

Resource Workshop interprets "3 –1" as an expression that evaluates to 2. The IBM Resource compiler interprets "3 –1" as two separate fields. If you add a comma after the first number, both compilers interpret the statement correctly:

```
ICON   3, -1, 10, 10, 0, 0
```

Floating operators in expressions

Resource Workshop's expression parser does not allow "floating" operators in constant expressions; the IBM Resource Compiler does. For example, the following expression is flagged as an error in Resource Workshop:

```
WS_SYSMENU | WS_CAPTION |
```

To correct the error, remove the last bitwise OR operator:

```
WS_SYSMENU | WS_CAPTION
```

Missing operators in expressions

Resource Workshop's expression parser requires that all operators required for an expression be present. The IBM Resource Compiler assumes that a missing operator is a bitwise OR operator. For example, the following expression is flagged as an error in Resource Workshop:

```
WS_SYSMENU  WS_CAPTION
```

To correct the error, add the bitwise OR operator:

```
WS_SYSMENU | WS_CAPTION
```

Macros in include directives

Resource Workshop doesn't support macro expansion in include directives. For example, the following fragment causes a compile error:

```
#define MYFILE "afile.h"
#include MYFILE
```

A P P E N D I X B

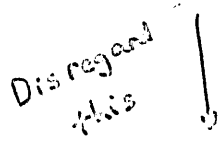
Borland PM Custom Controls

The Borland PM Custom Controls (BPMCC) library contains a custom dialog class and a set of custom dialog controls (buttons, check boxes, group shading boxes, and the like). BPMCC adds to the visual impact of your dialog boxes and optimizes their functionality.

Two of the online files included with Resource Workshop provide additional information about BPMCC:

- BPMCCAPI.RW provides technical information about the BPMCC application program interface.
- BPMCCSTYL.RW provides some style suggestions for designing Borland-style dialog boxes.

Using the Borland custom dialog class



The custom dialog class, BORDLG, works on both a visual and a functional level:

- It improves the appearance of your dialog window by painting the background with a brush that varies according to the target display device. For screens of VGA and higher resolution, the background is a fine grid of perpendicular white lines, giving the effect of "chiseled steel." For EGA and monochrome screens, the background is white.
- It optimizes the drawing of dialog boxes by calling the custom control drawing routines directly instead of waiting for PM to paint the controls. This eliminates the typically sluggish drawing of dialog boxes.

To use the custom dialog class,

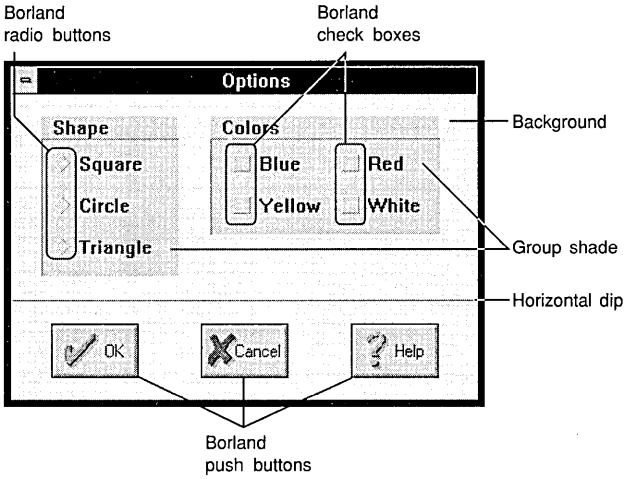
1. Open the dialog resource you want to convert.
2. Double-click the title bar of the dialog to display the Frame Style dialog box.
3. Enter "bordlg" as the Class and click OK.

Using Borland controls

Borland controls add a three-dimensional effect to your dialog boxes and give them more visual impact. To the end-user, they appear to function in the same manner as the standard PM controls, although they include several technical enhancements (described later).

The following figure shows a dialog box converted to BPMCC. It uses several Borland controls.

Figure B.1
Dialog box with Borland controls



The following list briefly describes each Borland control and shows its corresponding tool icon. As with standard PM controls, you can insert Borland controls in your dialogs by picking them from the Tools palette in the Dialog editor.

The description of each control includes its class. To see the class and other settings of any of these controls, display the Generic Control Style dialog box by holding down the *Ctrl* key and double-clicking on the control.

Group shade — A shaded rectangular box that groups other controls visually. It can appear recessed into the dialog box or raised above its surface. Its class is *BorShade*.

Horizontal dip — A horizontal dividing line that gives the impression of being etched into the surface of the dialog box. (You can convert a dip to a bump that appears to be raised above the surface of the dialog box.) Its class is *BorShade*.

Vertical dip — Same as horizontal dip, except it's vertical. Its class is *BorShade*.

Borland push button

A family of push buttons with symbols that have high visual impact, plus an owner-draw option. The Borland push buttons are larger than most standard PM push buttons. Their class is *BorBtn*.

Borland radio button

A raised, diamond-shaped radio button. When the button is clicked, a black diamond appears in its center and the button shading reverses, giving the impression that the button has been pushed down. There is also an owner-draw option. Its class is *BorRadio*.

Borland check box

A raised check box that displays a check mark instead of an "X." There is also an owner-draw option. Its class is *BorCheck*.

Button and check box enchancements

The Borland push buttons, radio buttons, and check boxes have the following functional enhancements over standard PM controls:

- An additional level of parent window notification and control over keyboard focus and tab movement. If you choose the Parent Notify option in the control's style dialog box, the control sends the appropriate messages from the following list at run time:
  - BBN_SETFOCUS indicates to the parent window that the push button, radio button, or check box has gained keyboard focus through an action other than a mouse click.
  - BBN_SETFOCUSMOUSE indicates to the parent window that the push button, radio button, or check box has gained keyboard focus through a mouse click.
  - BBN_GOTATAB indicates to the parent window that the user has pressed the Tab key while the push button, radio button, or check box has keyboard focus. The parent can intervene in the processing of the keystroke by returning a nonzero value.
  - BBN_GOTABTAB indicates to the parent window that the user has pressed Shift-Tab (back-tab) while the push button, radio button, or check box has keyboard focus. The parent can intervene in the processing of the keystroke by returning a nonzero value.
- An owner-draw option that allows the parent window to draw the push button, radio button, or check box. Because your application handles drawing the control, it won't necessarily look like a Borland control, but it will have the standard behavior of that class of control.

Using the BPMCC style dialog boxes

Four dialog boxes set the style of the BPMCC controls:

- Borland Button Style
- Borland Radio Button Style
- Borland Check Box Style
- Borland Shade Style

To display one of the Style dialog boxes, double-click on the control whose style you want to set.

Each has a control window for entering a caption and a control ID. The button style, radio button style, and check box style dialog boxes have Attributes options for Tab Stop, Disabled, Group, Visible, and Border, as well as Parent Notify and Owner Draw (described earlier in this appendix).

The next four sections describe the features unique to each of the style dialog boxes.

Borland Button Style dialog box

This dialog box lets you choose from the three button types: Pushbutton, Defpushbutton, and Bitmap.

Pushbutton and Defpushbutton

By default, Pushbutton is the selected option. A Defpushbutton has a bold border to identify it to the end-user as the *default button*, which is executed when the user presses the Enter key.

When you first place a Borland button in your dialog box, it takes the next available control ID. To change the button to one of the standard Borland buttons, change the control ID to one of the preset values in the following table:

Table B.1
Predefined BPMCC button controls

ID value	Type	Image
1	OK	Green check mark
2	Cancel	Red X
3	Abort	Panic button
4	Retry	Slot machine
5	Ignore	55 mph speed-limit sign
6	Yes	Green check mark
7	No	Red circle and slash
8	Enter	Green check mark
998	Help	Blue question mark

Bitmap

If you choose the Bitmap option, you can insert a bitmap image (based on its control ID) into the button. To read in a bitmap:

1. Switch to the Bitmap Editor and create a bitmap image. (See Chapter 5 for information about creating bitmaps.)
2. In the Bitmap Editor, choose Resource | Rename to display the Rename Resource dialog box and then do either of the following:
   - In the New Name text box, enter an integer value that equals the control ID of the button plus the appropriate offset from Table B.2.
   - Rename the bitmap and then assign it an identifier whose value equals the control ID of the button plus the appropriate offset from Table B.2. (Creating identifiers is described in Chapter 3.)
3. Close the Bitmap Editor.
4. Return to the Dialog Editor. If the bitmap doesn't immediately appear in the BPMCC button, resize the button. The bitmap should then appear.

The bitmap won't display in the Dialog editor until you close the Bitmap Editor.

Table B.2
Bitmap offsets

Button state	Offset for VGA/higher	Offset for EGA/monochrome
Standard	1000	2000
Pressed	3000	4000
Keyboard focus	5000	6000

For example, to display the keyboard focus bitmap for a button whose control ID is 276, enter 5276 for a VGA system or 6276 for an EGA system.

Borland Radio Button Style dialog box

This dialog box lists two button styles:

- *Radio button*. Highlighting and deselection don't happen automatically. The application must call the **CheckRadioButton** function to send a BM_SETCHECK message to highlight the selected button and deselect the other buttons.
- *Auto radio button*. BPMCC and PM combine to handle highlighting the selected button and deselecting the other buttons. This is the default option.

Borland Check Box Style dialog box

This dialog box lists four check box styles:

- *Check box*. The box is not checked automatically. The application must call the **CheckDlgButton** function to send a BM_SETCHECK message to check the selected box.

- *Auto check box.* BPMCC and PM combine to handle checking the selected box. This is the default option.
- *3-state.* The box is not checked automatically. The application must call the **CheckDlgButton** function to send a BM_SETCHECK message to check the selected box.

  The button's three states are on, off, and "indeterminate," which is displayed as a checkerboard pattern. The application determines what is meant by "indeterminate."
- *Auto 3-state.* BPMCC and PM combine to handle checking the selected box.

Borland Shade Style dialog box

This dialog box sets the style for controls you add with any of these three tools: Group Shade, Horizontal Dip, and Vertical Dip. Using the Shade Style radio buttons, you can make the following conversions:

- *Group Shade to Raised Shade.* Group shades and raised shades are used to enclose controls with related functions—like radio buttons and check boxes. Group shades appear recessed below the surface of the dialog box; raised shades appear raised above the surface.
- *Horizontal Dip to Horizontal Bump, Vertical Dip to Vertical Bump.* Dips are intended to act as separators in the dialog box background or in raised shades; bumps are intended as separators in recessed gray shade boxes.

Borland Static Text Style dialog box

Use this dialog box to enter the text and set attributes and control style for Borland static text.

In addition to the standard attributes (Disabled, Group, and Visible), static text has two additional attributes.

- When the Border option is checked, the static text is surrounded by a standard PM border that uses the current color for the Window Frame (see the Windows Control Panel).
- When the No Underline option is checked, an ampersand (&) appears as a literal character, instead of underlining the next character.

Modifying existing applications for BPMCC

Resource Workshop lets you modify existing PM applications with Borland-style custom controls (3D buttons, dialog boxes with the "chiseled steel" look, and so on).

Using BPMCC in C and C++ programs

The easiest way to use BPMCC is to click on the BPMCC icon provided on by the PM.

If you don't use the BWCC icon, you must do all of the following:

- Add a **#include** for BPMCC.H to your .C or .CPP file.
- Link the BPMCC.LIB to your C or C++ files and libraries.

Tips on editing resources

This section discusses considerations to keep in mind when editing resources of existing applications.

- **Accelerators** If you add an accelerator, make sure it returns the same ID value as its corresponding menu command. If you don't, the accelerator will either execute the wrong command or do nothing.
- **Bitmaps, cursors, and icons** You can modify existing bitmaps, cursors, and icons. Don't delete bitmaps, cursors, or icons, and don't try to add new ones. In most cases the application won't be able to use them.
- **Dialog boxes** You can reposition items in a dialog box and convert controls to their Borland custom control counterparts. As you edit, be sure not to change the type of control associated with each control ID value. For example, if control ID 100 is a check box, don't change it to a radio button, because the application will still treat it as a check box.

  In most cases you can remove controls that aren't directly tied to the application's functionality. For example, you can usually remove a caption, a static text item that has no effect on how the application works. Don't remove an edit control; it *does* affect how the application works. Don't add new controls; the application won't be able to use them.
- **Menus** With most applications, you can safely move commands within a menu. Don't, however, move commands from one menu to another. (For example, don't move the Open command from the File menu to the Edit menu.) If you do, the application might be unable to display context-sensitive Help or to check or uncheck the menu commands. Never change the order of the menus in the menu bar. For example, if File is the first menu, don't make it the second.
- **String tables** Use caution when editing existing string tables. Some programs load the strings into buffers of fixed size, and adding text to an existing string could cause the buffer to overflow. Don't add new strings; the application won't be able to use them.

Index

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

Borland

Corporate Headquarters: 100 Borland Way, Scotts Valley, CA 95066-3249, (408) 431-1000. Offices in: Australia, Belgium, Canada, Chile, Denmark, France, Germany, Hong Kong, Italy, Japan, Korea, Latin America, Malaysia, Netherlands, New Zealand, Singapore, Spain, Sweden, Taiwan, and United Kingdom • Part # BCP1415WW21775 • BOR 7005