Microsoft® Windows™ 3.1

Programmer's Reference

Volume 4

Resources

Microsoft® Windows™ 3.1

Programmer's Reference

Volume 4

Resources

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Windows programmer's reference.
p. cm.
Includes indexes.
Contents: v. 1. Overview -- v. 2. Functions -- v. 3. Messages, structures, macros -- v. 4. Resources.
ISBN 1-55615-453-4 (v. 1). -- ISBN 1-55615-463-1 (v. 2). -- ISBN 1-55615-464-X (v. 3). -- ISBN 1-55615-494-1 (v. 4)
1. Microsoft Windows (Computer program) I. Microsoft Corporation.
QA76.76.W56M532 1992
005.4'3--dc20 91-34199
CIP

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 MLML 7 6 5 4 3 2

Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation.

Distributed to the book trade outside the United States and Canada by Penguin Books Ltd.

Penguin Books Ltd., Harmondsworth, Middlesex, England
Penguin Books Australia Ltd., Ringwood, Victoria, Australia
Penguin Books N.Z. Ltd., 182-190 Wairau Road, Auckland 10, New Zealand

British Cataloging-in-Publication Data available.

Document No. PC30211-0492

Introduction

The Microsoft® Windows™ operating system is a single-user personal-computer operating system that employs a graphical user interface. This graphical interface uses a variety of resources that must be constructed in specific formats. This manual, the *Microsoft Windows Programmer's Reference, Volume 4*, describes these resource formats and executable-file headers.

Part 1, "File Formats," describes the formats for the principal types of files used by Windows applications. The chapters in this part provide detailed information about the file formats, as well as about the MS-DOS® and Windows executable-file headers and resource formats within executable files. Topics include the formats for the following types of files: graphics, clipboard, font, group, calendar, object-module, library, symbol, and metafile.

Part 2, "Tools Reference," provides detailed reference information about the statements, commands, and macros for tools used to create and maintain Windows resources. Topics include resource-definition statements, assembly-language macros, and help statements and macros. Each entry in this section gives the purpose of the command or macro; its complete syntax, parameters, and return values; and cross-references to related commands or macros. Many entries also include expanded comments on the use of the command or macro.

How to Use This Manual

This manual describes the Windows resource-file formats in individual chapters. Each chapter describes the format that should be used for the type of file associated with a specific resource or activity. For example, the chapter on graphics file formats describes the formats used with bitmap, icon, and cursor resource files.

Each chapter has two parts: a general description of the file type and a detailed presentation of the format. Chapters in Part 2, "Tools Reference," describe only the file format and not the tool. For more information about the associated tools, see *Microsoft Windows Programming Tools*.

Document Conventions

The following conventions are used throughout this manual to define syntax:

Convention	Meaning
Bold text	Denotes a term or character to be typed literally, such as a resource-definition statement or function name (**MENU** or **CreateWindow**), an MS-DOS command, or a command-line option (**/nod**). You must type these terms exactly as shown.
Italic text	Denotes a placeholder or variable: You must provide the actual value. For example, the statement **SetCursorPos**(*X*,*Y*) requires you to substitute values for the *X* and *Y* parameters.
[ ]	Enclose optional parameters.
|	Separates an either/or choice.
...	Specifies that the preceding item may be repeated.

`BEGIN`
`.`
`.`
`.`
`END`
	Represents an omitted portion of a sample application.

In addition, certain text conventions are used to help you understand this material:

Convention	Meaning
SMALL CAPITALS	Indicate the names of keys, key sequences, and key combinations—for example, ALT+SPACEBAR.
FULL CAPITALS	Indicate filenames and paths, most type and structure names (which are also bold), and constants.
`monospace`	Sets off code examples and shows syntax spacing.

Contents

Part 2 Tools Reference

File Formats

Part 1

Graphics File Formats

Chapter 1

This chapter describes the graphics-file formats used by the Microsoft Windows operating system. Graphics files include bitmap files, icon-resource files, and cursor-resource files.

1.1 Bitmap-File Formats

Windows bitmap files are stored in a device-independent bitmap (DIB) format that allows Windows to display the bitmap on any type of display device. The term "device independent" means that the bitmap specifies pixel color in a form independent of the method used by a display to represent color. The default filename extension of a Windows DIB file is .BMP.

1.1.1 Bitmap-File Structures

Each bitmap file contains a bitmap-file header, a bitmap-information header, a color table, and an array of bytes that defines the bitmap bits. The file has the following form:

```
BITMAPFILEHEADER bmfh;
BITMAPINFOHEADER bmih;
RGBQUAD          aColors[];
BYTE             aBitmapBits[];
```

The bitmap-file header contains information about the type, size, and layout of a device-independent bitmap file. The header is defined as a **BITMAPFILEHEADER** structure.

The bitmap-information header, defined as a **BITMAPINFOHEADER** structure, specifies the dimensions, compression type, and color format for the bitmap.

The color table, defined as an array of **RGBQUAD** structures, contains as many elements as there are colors in the bitmap. The color table is not present for bitmaps with 24 color bits because each pixel is represented by 24-bit red-green-blue (RGB) values in the actual bitmap data area. The colors in the table should appear in order of importance. This helps a display driver render a bitmap on a device that cannot display as many colors as there are in the bitmap. If the DIB is in Windows version 3.0 or later format, the driver can use the **biClrImportant** member of the **BITMAPINFOHEADER** structure to determine which colors are important.

The **BITMAPINFO** structure can be used to represent a combined bitmap-information header and color table.

The bitmap bits, immediately following the color table, consist of an array of **BYTE** values representing consecutive rows, or "scan lines," of the bitmap. Each scan line consists of consecutive bytes representing the pixels in the scan line, in left-to-right order. The number of bytes representing a scan line depends on the

color format and the width, in pixels, of the bitmap. If necessary, a scan line must be zero-padded to end on a 32-bit boundary. However, segment boundaries can appear anywhere in the bitmap. The scan lines in the bitmap are stored from bottom up. This means that the first byte in the array represents the pixels in the lower-left corner of the bitmap and the last byte represents the pixels in the upper-right corner.

The **biBitCount** member of the **BITMAPINFOHEADER** structure determines the number of bits that define each pixel and the maximum number of colors in the bitmap. These members can have any of the following values:

Value	Meaning
1	Bitmap is monochrome and the color table contains two entries. Each bit in the bitmap array represents a pixel. If the bit is clear, the pixel is displayed with the color of the first entry in the color table. If the bit is set, the pixel has the color of the second entry in the table.
4	Bitmap has a maximum of 16 colors. Each pixel in the bitmap is represented by a 4-bit index into the color table. For example, if the first byte in the bitmap is 0x1F, the byte represents two pixels. The first pixel contains the color in the second table entry, and the second pixel contains the color in the sixteenth table entry.
8	Bitmap has a maximum of 256 colors. Each pixel in the bitmap is represented by a 1-byte index into the color table. For example, if the first byte in the bitmap is 0x1F, the first pixel has the color of the thirty-second table entry.
24	Bitmap has a maximum of 2^{24} colors. The **bmiColors** (or **bmciColors**) member is NULL, and each 3-byte sequence in the bitmap array represents the relative intensities of red, green, and blue, respectively, for a pixel.

The **biClrUsed** member of the **BITMAPINFOHEADER** structure specifies the number of color indexes in the color table actually used by the bitmap. If the **biClrUsed** member is set to zero, the bitmap uses the maximum number of colors corresponding to the value of the **biBitCount** member.

An alternative form of bitmap file uses the **BITMAPCOREINFO**, **BITMAPCOREHEADER**, and **RGBTRIPLE** structures.

For a full description of the bitmap structures, see the *Microsoft Windows Programmer's Reference, Volume 3*.

1.1.2 Bitmap Compression

Windows versions 3.0 and later support run-length encoded (RLE) formats for compressing bitmaps that use 4 bits per pixel and 8 bits per pixel. Compression reduces the disk and memory storage required for a bitmap.

1.1.2.1 Compression of 8-Bits-per-Pixel Bitmaps

When the **biCompression** member of the **BITMAPINFOHEADER** structure is set to BI_RLE8, the DIB is compressed using a run-length encoded format for a 256-color bitmap. This format uses two modes: encoded mode and absolute mode. Both modes can occur anywhere throughout a single bitmap.

Encoded Mode A unit of information in encoded mode consists of two bytes. The first byte specifies the number of consecutive pixels to be drawn using the color index contained in the second byte.

The first byte of the pair can be set to zero to indicate an escape that denotes the end of a line, the end of the bitmap, or a delta. The interpretation of the escape depends on the value of the second byte of the pair, which must be in the range 0x00 through 0x02. Following are the meanings of the escape values that can be used in the second byte:

Second byte	Meaning
0	End of line.
1	End of bitmap.
2	Delta. The two bytes following the escape contain unsigned values indicating the horizontal and vertical offsets of the next pixel from the current position.

Absolute Mode Absolute mode is signaled by the first byte in the pair being set to zero and the second byte to a value between 0x03 and 0xFF. The second byte represents the number of bytes that follow, each of which contains the color index of a single pixel. Each run must be aligned on a word boundary.

Following is an example of an 8-bit RLE bitmap (the two-digit hexadecimal values in the second column represent a color index for a single pixel):

Compressed data	Expanded data
03 04	04 04 04
05 06	06 06 06 06 06
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	Move 5 right and 1 down
02 78	78 78
00 00	End of line
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	End of RLE bitmap