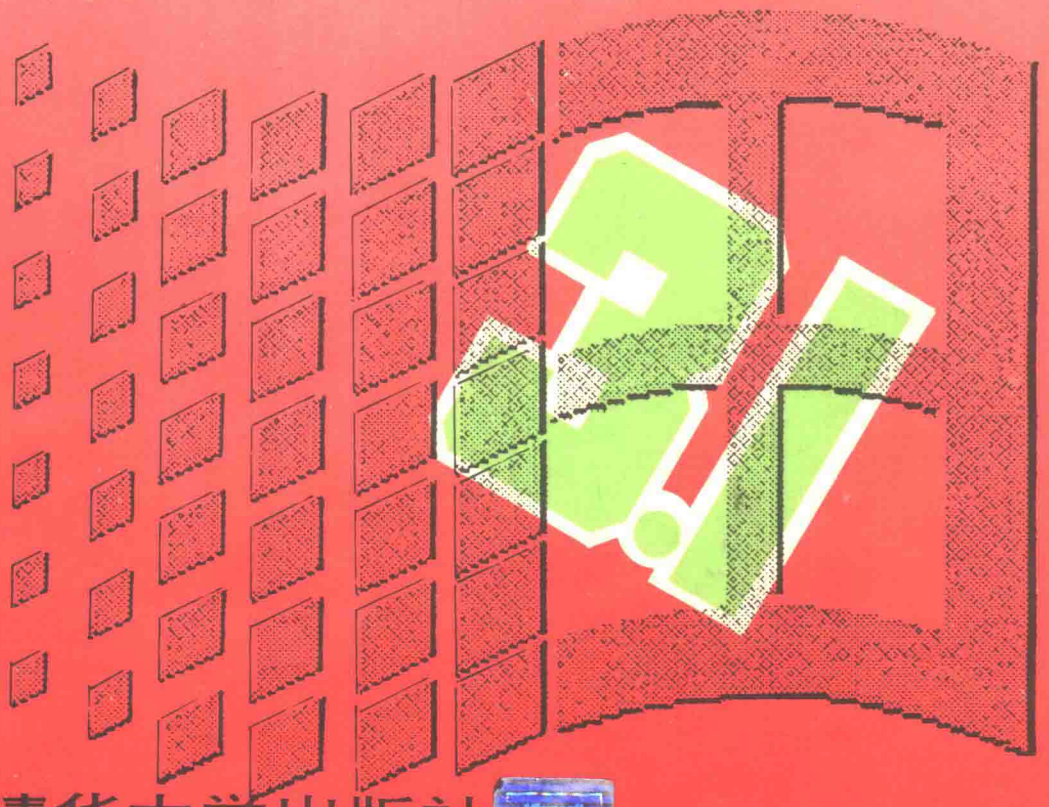
Microsoft Microsoft Microsoft

Microsoft Windows 3.1

程序员参考大全(三)

—消息 数据结构和宏

蒋维杜 全 正 张军
史开宇 李 逸 译



清华大学出版社

Microsoft

Windows 3.1

程序员参考大全(三)

—消息、数据结构和宏

[美] Microsoft 公司 著

蒋维杜 全 正 张 军 译

史开宇 李 逸

清华大学出版社

Microsoft Windows 3.1 Programmer's Reference
Volume 3 Messages, Structures, and Macros

Adobe®和 PostScript®是 Adobe Systems 公司的注册商标。
Apple®、True Type®和 Macintosh®是 Apple Computer 公司的注册商标。
PANOSE™是 ElseWare 公司的注册商标。
Epson®和 FX®是 Epson America 公司的注册商标。
Hewlett-Packard®、HP®、LaserJet®和 PCL®是 Hewlett-Packard 公司的注册商标。
IBM®是 International Business Machines 公司的注册商标。
ITC Zapf Chancery 和 ITC Zapf Dingbats®是 International Typeface 公司的注册商标。
Helvetica®、New Century Schoolbook®、Palatino®、Times®和 Times Roman 是 Linotype AG 或它下属的注册商标。
CodeView®、Microsoft®、MS®、MS-DOS®和 QuickC®是 Microsoft 公司的注册商标,QuickBasic™和 Windows™是 Microsoft 公司的商标。
Arial 和 Times New Roman®是 Monotype Corporation PLC 公司的注册商标。
Okidata 是 Oki America 公司的注册商标。

(京)新登字 158 号

Microsoft Windows 3.1 程序员参考大全(三)
——消息、数据结构和宏
[美] Microsoft 公司 著
蒋维杜 全 正 张 军 史开宇 李 逸 译
☆
清华大学出版社出版
北京 清华园
清华大学印刷厂印刷
新华书店总店科技发行所发行
☆
开本:787×1092 1/16 印张:25.75 字数:610 千字
1993 年 7 月第 1 版 1993 年 7 月第 1 次印刷
印数:0001—5000
ISBN 7-302-01223-7/TP·461
定价:39.00 元

引 言

本书描述了由 Microsoft® Windows™操作系统所支持的数据类型、消息、结构、宏和打印机指令，以及动态数据交换(DDE)事务、File Manager 事件、光栅操作码、虚键码和字符表。

本书的组织

以下是对本书各章及附录的概述：

- 第 1 章“数据类型”描述了用来定义 Windows 程序设计界面(API)中有关参数、返回值的大小和意义的关键字。
- 第 2 章“消息”，描述了 Windows 操作系统用于和应用程序进行通信的格式化的窗口消息，还描述了把发生在控制内的动作通知给控制的父窗口的通知消息。
- 第 3 章“结构”，定义了 Windows API 中函数部分有关的数据结构。
- 第 4 章“宏”，描述了宏的目的，定义了宏的参数，这些宏有助于在 Windows 应用程序中管理数据。
- 第 5 章“打印机指令”，列出 Windows 操作系统所用的所有打印机指令。
- 第 6 章“动态数据交换事务”，描述了由动态数据交换管理库(DDEML)发送给应用程序的动态数据交换(DDE)回调函数的事务，这些事务把影响应用程序的 DDE 活动通知应用程序。
- 第 7 章“File Manager 事件与消息”，提供了 File Manager 发送的用于和 File Manager 扩展动态链接库(DLL)进行通信的事件及菜单命令的描述；这一章还描述了 DLL 向 File Manager 发送的用于检索信息的消息。
- 第 8 章“Control Panel 消息”，列出了由 Control Panel 发送的用于和 Control Panel DLL 进行通信的消息。
- 第 9 章“公共对话框消息”描述公共对话框能够发送消息给应用程序，这些消息通知应用程序用户在对话框中已经做或改变了选择。
- 第 10 章“可安装驱动程序消息”，列出了 Windows 操作系统通知可安装驱动程序特定事件的消息。
- 附录 A“二元和三元光栅操作码”，列表并描述了图形设备接口(GDI)使用的所有二元、三元光栅操作。
- 附录 B“虚键码”，给出了 Windows 虚键码和符号常量名、十六进制值和对应键盘键。
- 附录 C“字符表”，说明了 Windows 操作系统使用的 Windows 字符集、符号字符集和 OEM 字符集。

文本规则

本书用下列规则定义语法：

规　则	意　义
正体字	指示逐字输入一个词或一个字符，如资源定义语句或函数名(MENU 或 Create Window)，MS-DOS 命令或命令行选择项(/nod)。编程时须严格按所示内容编写。
斜体字	指示一个占位符或变量：编程时提供实际值。例如，语句 Set Cursor Pos(*X*, *Y*)要求用值替代 *X* 和 *Y* 参数。
[ ]	括起选择项。
|	分隔"二选一"或者"或"选项。
…	指示重复前面内容。
BEGIN	
:	表示范例程序中的省略内容。
END	

此外，为帮助学习本书，书中采用下列文字规则：

规　则	意　义
"小号"大写字符	表示键名、键序列或键组合——例如，ALT+SPACEBAR。
"大号"大写字符	表示文件名和路径，类型和结构名(也可能是黑体字)，常量。
单空格	分隔代码及语法。

目 录

第 1 章

数 据 类 型

本章中的数据类型是一些关键字，这些关键字定义了 Microsoft Windows 3.1 版本操作系统的函数中有关参数和返回值的大小和意义。下列表中包括了字符、整数和布尔类型；指针类型以及句柄。其中的字符、整数和布尔类型与多数 C 编译器是一致的；多数指针类型的名称以 P，N（表示近指针），或 LP（表示远指针）作为前缀打头。近指针只能访问当前数据段内的数据，而一个远指针可以访问一个 32 位的数据段：偏移量的地址。Windows 应用程序中利用句柄来访问已经被装入内存的资源。Windows 通过一些内部维护的表来支持对这些资源的访问，这些表分别为每个句柄设置一项。句柄表中的每一项包含地址和标识这个资源类型的方式。

下列表中按字母顺序定义了 Windows 的数据类型：

类　型	定　义
ABORTPROC	指向一个回调函数 AbortProc 的 32 位指针。
ATOM	用作一个原子句柄的 16 位值。
BOOL	16 位布尔值。
BYTE	8 位无符号整数。利用 LPBYTE 创建 32 位指针；利用 PBYTE 创建匹配编译器内存模式的指针。
CATCHBUF[9]	用于 Catch 函数的 18 字节缓冲区。
COLORREF	用作颜色值的 32 位值。
DLGPROC	指向一个对话框过程的 32 位指针。
DWORD	32 位无符号整数或一个数据段：偏移量的地址。利用 LPDWORD 创建 32 位指针；利用 PDWORD 创建匹配编译器内存模式的指针。
FARPROC	指向一个函数的 32 位指针。
FNCALLBACK	标识 DdeCallback 函数的 32 位值。利用 PFNCALLBACK 创建匹配编译器内存模式的指针。
FONTENUMPROC	指向一个回调函数 EnumFontsProc 的 32 位指针。
GLOBALHANDLE	用作表示一个全局内存对象句柄的 16 位值。
GNOTIFYPROC	指向一个回调函数 NotifyProc 的 32 位指针。
GOBJENUMPROC	指向一个回调函数 EnumObjectsProc 的 32 位指针。
GRAYSTRINGPROC	指向一个回调函数 GrayStringProc 的 32 位指针。
HANDLE	用作一个通用句柄的 16 位值。利用 LPHANDLE 创建 32 位指针；利用 SPHANDLE 创建 16 位指针；利用 PHANDLE 创建匹配编译器内存模式的指针。
HCURSOR	用作一个光标句柄的 16 位值。
HFILE	用作一个文件句柄的 16 位值。
HGDIOBJ	用作一个图形设备接口（GDI）对象句柄的 16 位值。

（续表）

类　型	定　义
HGLOBAL	用作表示一个全局内存对象句柄的 16 位值。
HHOOK	用作一个钩子句柄的 32 位值。
HKEY	用作表示注册数据库中一个关键字句柄的 32 位值。利用 PHKEY 创建 32 位指针。
HLOCAL	用作表示一个局部内存对象句柄的 16 位值。
HMODULE	用作一个模块句柄的 16 位值。
HOBJECT	用作表示一个 OLE 对象句柄的 16 位值。
HWND	用作表示一个窗口句柄的 16 位值。
HOOKPROC	指向一个钩子函数的 32 位指针。
HRSRC	用作一个资源句柄的 16 位值。
LHCLIENTDOC	用作表示一个 OLE 用户文本句柄的 32 位值。
LHSERVER	用作表示一个 OLE 服务器句柄的 32 位值。
LHSERVERDOC	用作表示一个 OLE 服务器文本句柄的 32 位值。
LINEDDAPROC	指向一个回调函数 LineDDAProc 的 32 位指针。
LOCALHANDLE	用作表示一个局部内存对象句柄的 16 位值。
LONG	32 位带符号整数。
LPABC	指向一个 ABC 结构的 32 位指针。
LPARAM	作为一个参数传递给一个窗口过程或回调函数的 32 位带符号值。
LPBI	指向一个 BANDINFOSTRUCT 结构的 32 位指针。
LPBITMAP	指向一个 BITMAP 结构的 32 位指针。利用 NPBITMAP 创建 16 位指针；利用 PBITMAP 创建匹配编译器内存模式的指针。
LPBITMAPCOREHEADER	指向一个 BITMAPCOREHEADER 结构的 32 位指针。利用 PBITMAPCOREHEADER 创建匹配编译器内存模式的指针。
LPBITMAPCOREINFO	指向一个 BITMAPCOREINFO 结构的 32 位指针。利用 PBITMAP-COREINFO 创建匹配编译器内存模式的指针。
LPBITMAPFILEHEADER	指向一个 BITMAPFILEHEADER 结构的 32 位指针。利用 PBITMAPFILEHEADER 创建匹配编译器内存模式的指针。
LPBITMAPINFO	指向一个 BITMAPINFO 结构的 32 位指针。利用 PBITMARINFO 创建匹配编译器内存模式的指针。
LPBITMAPINFOHEADER	指向一个 BITMAPINFOHEADER 结构的 32 位指针。利用 PBITMAPINFOHEADER 创建匹配编译器内存模式的指针。
LPCATCHBUF	指向一个 CATCHBUF 数组的 32 位指针。
LPCBT_CREATEWND	指向一个 CBT_CREATEWND 结构的 32 位指针。
LPCHOOSECOLOR	指向一个 CHOOSECOLOR 结构的 32 位指针。
LPCHOOSEFONT	指向一个 CHOOSEFONT 结构的 32 位指针。
LPCLIENTCREATESTRUCT	指向一个 CLIENTCREATESTRUCT 结构的 32 位指针。
LPCOMPAREITEMSTRUCT	指向一个 COMPAREITEMSTRUCT 结构的 32 位指针，利用 PCOMPAREITEMSTRUCT 创建匹配编译器内存模式的指针。
LPCPLINFO	指向一个 CPLINFO 结构的 32 位指针。利用 PCPLINFO 创建匹配编译器内存模式的指针。
LPCREATESTRUCT	指向一个 CREATESTRUCT 结构的 32 位指针。

（续表）

类　型	定　义
LPCSTR	指向一个不可修改字符串的 32 位指针。
LPCTLINFO	指向一个 CTLINFO 结构的 32 位指针。利用 PCTLINFO 创建匹配编译器内存模式的指针。
LPCTLSTYLE	指向一个 CTLSTYLE 结构的指针。利用 PCTLSTYLE 创建匹配编译器内存模式的指针。
LPDCB	指向一个 DCB 结构的 32 位指针。
LPDEBUGHOOKINFO	指向一个 DEBUGHOOKINFO 结构的 32 位指针。
LPDELETEITEMSTRUCT	指向一个 DELETEITEMSTRUCT 结构的 32 位指针。利用 PDELETEITEMSTRUCT 创建匹配编译器内存模式的指针。
LPDEVMODE	指向一个 DEVMODE 结构的 32 位指针。利用 NPDEVMODE 创建 16 位指针；利用 PDEVMODE 创建匹配编译器内存模式的指针。
LPDEVNAMES	指向一个 DEVNAMES 结构的 32 位指针。
LPDOCINFO	指向一个 DOCINFO 结构的 32 位指针。
LPDRAWITEMSTRUCT	指向一个 DRAWITEMSTRUCT 结构的 32 位指针。利用 PDRAWITEMSTRUCT 创建匹配编译器内存模式的指针。
LPDRIVERINFOSTRUCT	指向一个 DRIVERINFOSTRUCT 结构的 32 位指针。
LPDRVCONFIGINFO	指向一个 DRVCONFIGINFO 结构的 32 位指针。利用 PDRVCONFIGINFO 创建匹配编译器内存模式的指针。
LPEVENTMSG	指向一个 EVENTMSG 结构的 32 位指针。利用 NPEVENTMSG 创建 16 位指针；利用 PEVENTMSG 创建匹配编译器内存模式的指针。
LPDRIVERINFOSTRUCT	指向一个 DRIVERINFOSTRUCT 结构的 32 位指针。
LPFINDREPLACE	指向一个 FINDREPLACE 结构的 32 位指针。
LPFMS_GETDRIVEINFO	指向一个 FMS_GETDRIVEINFO 结构的 32 位指针。
LPFMS_GETFILESEL	指向一个 FMS_GETFILESEL 结构的 32 位指针。
LPFMS_LOAD	指向一个 FMS_LOAD 结构的 32 位指针。
LPHANDLETABLE	指向一个 HANDLETABLE 结构的 32 位指针。利用 PHANDLETABLE 创建匹配编译器内存模式的指针
LPHELPWININFO	指向一个 HELPWININFO 结构的 32 位指针。利用 PHELPWININFO 创建匹配编译器内存模式的指针。
LPINT	指向一个 16 位的带符号的值的 32 位指针。利用 PINT 创建匹配编译器内存模式的指针。
LPKERNINGPAIR	指向一个 KERNINGPAIR 结构的 32 位指针。
LPLOGBRUSH	指向一个 LOGBRUSH 结构的 32 位指针。利用 NPLOGBRUSH 创建 16 位指针；利用 PLOGBRUSH 创建匹配编译器内存模式的指针。
LPLOGFONT	指向一个 LOGFONT 结构的 32 位指针。利用 NPLOGFONT 创建 16 位指针；利用 PLOGFONT 创建匹配编译器内存模式的指针。
LPLOGPALETTE	指向一个 LOGPALETTE 结构的 32 位指针。利用 NPLOGPALETTE 创建 16 位指针；利用 PLOGPALETTE 创建匹配编译器内存模式的指针。
LPLOGPEN	指向一个 LOGPEN 结构的 32 位指针。利用 NPLOGPEN 创建 16 位指针；利用 PLOGPEN 创建匹配编译器内存模式的指针。

（续表）

类　型	定　义
LPLONG	指向一个 32 位带符号整数的 32 位指针。利用 PLONG 创建匹配编译器内存模式的指针。
LPMAT2	指向一个 MAT2 结构的 32 位指针。
LPMDICREATESTRUCT	指向一个 MDICREATESTRUCT 结构的 32 位指针。
LPMEASUREITEMSTRUCT	指向一个 MEASUREITEMSTRUCT 结构的 32 位指针。利用 PMEASUREITEMSTRUCT 创建匹配编译器内存模式的指针。
LPMETAFILEPICT	指向一个 METAFILEPICT 结构的 32 位指针。
LPMETARECORD	指向一个 METARECORD 结构的 32 位指针，利用 PMETARECORD 创建匹配编译器内存模式的指针。
LPMOUSEHOOKSTRUCT	指向一个 MOUSEHOOKSTRUCT 结构的 32 位指针。
LPMSG	指向一个 MSG 结构的 32 位指针。利用 NPMSG 创建 16 位指针；利用 PMSG 创建匹配编译器内存模式的指针。
LPNCCALCSIZE_PARAMS	指向一个 NCCALCSIZE_PARAMS 结构的 32 位指针。
LPNEWCPLINFO	指向一个 NEWCPLINFO 结构的 32 位指针。利用 PNEWCPLINFO 创建匹配编译器内存模式的指针。
LPNEWTEXTMETRIC	指向一个 NEWTEXTMETRIC 结构的 32 位指针。利用 NPNEWTEXTMETRIC 创建 16 位指针；利用 PNEWTEXTMETRIC 创建匹配编译器内存模式的指针。
LPOFSTRUCT	指向一个 OFSTRUCT 结构的 32 位指针。利用 NPOFSTRUCT 创建 16 位指针。利用 POFSTRUCT 创建匹配编译器内存模式的指针。
LPOLECLIENT	指向 OLECLIENT 结构的 32 位指针。
LPOLECLIENTVTBL	指向 OLECLIENTVTBL 结构的 32 位指针。
LPOLEOBJECT	指向 OLEOBJECT 结构的 32 位指针。
LPOLEOBJECTVTBL	指向 OLEOBJECTVTBL 结构的 32 位指针。
LPOLESERVER	指向 OLESERVER 结构的 32 位指针。
LPOLESERVERDOC	指向 OLESERVERDOC 结构的 32 位指针。
LPOLESERVERDOCVTBL	指向 OLESERVERDOCVTBL 结构的 32 位指针。
LPOLESERVERVTBL	指向 OLESERVERVTBL 结构的 32 位指针。
LPOLESTREAM	指向 OLESTREAM 结构的 32 位指针。
LPOLESTREAMVTBL	指向 OLESTREAMVTBL 结构的 32 位指针。
LPOLETARGETDEVICE	指向 OLETARGETDEVICE 结构的 32 位指针。
LPOPENFILENAME	指向 OPENFILENAME 结构的 32 位指针。
LPOUTLINETEXTMETRIC	指向一个 OUTLINETEXTMETRIC 结构的 32 位指针。
LPPAINTSTRUCT	指向一个 PAINTSTRUCT 结构的 32 位指针。利用 NPPAINTSTRUCT 创建 16 位指针；利用 PPAINTSTRUCT 创建匹配编译器内存模式的指针。
LPPALETTEENTRY	指向一个 PALETTEENTRY 结构的 32 位指针。
LPPOINT	指向一个 POINT 结构的 32 位指针。利用 NPPOINT 创建 16 位指针；利用 PPOINT 创建匹配编译器内存模式的指针。
LPPOINTFX	指向一个 POINTFX 结构的 32 位指针。
LPPRINTDLG	指向一个 PRINTDLG 结构的 32 位指针。

（续表）

类　型	定　义
LPRASTERIZER_STATUS	指向一个 RASTERIZER_STATUS 结构的 32 位的指针。
LPRECT	指向一个 RECT 结构的 32 位指针。利用 NPRECT 创建 16 位指针；利用 PRECT 创建匹配编译器内存模式的指针。
LPRGBQUAD	指向一个 RGBQUAD 结构的 32 位指针。
LPRGBTRIPLE	指向一个 RGBTRIPLE 结构的 32 位指针。
LPSEGINFO	指向一个 SEGINFO 结构的 32 位指针。
LPSIZE	指向一个 SIZE 结构的 32 位指针。利用 NPSIZE 创建 16 位指针；利用 PSIZE 创建匹配编译器内存模式的指针。
LPSTR	指向一个字符串的 32 位指针。利用 NPSTR 创建 16 位指针；利用 PSTR 创建匹配编译器内存模式的指针。
LPTEXTMETRIC	指向一个 TEXTMETRIC 结构的 32 位指针。利用 NPTEXTMETRIC 创建 16 位指针；利用 PTEXTMETRIC 创建匹配编译器内存模式的指针。
LPTTPOLYCURVE	指向一个 TTPOLYCURVE 结构的 32 位指针。
LPTTPOLYGONHEADER	指向一个 TTPOLYGONHEADER 结构的 32 位指针。
LPVOID	指向一个未说明类型的 32 位指针。
LPWINDOWPLACEMENT	指向一个 WINDOWPLACEMENT 结构的 32 位指针。利用 PWINDOWPLACEMENT 创建匹配编译器内存模式的指针。
LPWINDOWPOS	指向一个 WINDOWPOS 结构的 32 位指针。
LPWNDCLASS	指向一个 WNDCLASS 结构的 32 位指针。利用 NPWNDCLASS 创建 16 位指针；利用 PWNDCLASS 创建匹配编译器内存模式的指针。
LPWORD	指向一个 16 位无符号值的 32 位指针。利用 PWORD 创建匹配编译器内存模式的指针。
LRESULT	从一个窗口过程或回调函数返回的 32 位带符号值。
MFENUMPROC	指向一个回调函数 EnumMetaFileProc 的 32 位指针。
NEARPROC	指向一个函数的 16 位指针。
OLECLIPFORMAT	用作一个标准的裁剪板格式的 16 位值。
PATTERN	等同于 LOGBRUSH 结构。利用 LPPATTERN 创建 32 位指针；利用 NPPATTERN 创建 16 位指针；利用 PPATTERN 创建匹配编译器内存模式的指针。
PCONVCONTEXT	指向一个 CONVCONTEXT 结构的 32 位指针。
PCONVINFO	指向一个 CONVINFO 结构的 32 位指针。
PHSZPAIR	指向一个 HSZPAIR 结构的 32 位指针。
PROPENUMPROC	指向一个回调函数 EnumPropFixedProc 或 EnumPropMovableProc 的 32 位指针。
RSRCHDLRPROC	指向一个回调函数 LoadProc 的 32 位指针。
TIMERPROC	指向一个回调函数 TimerProc 的 32 位指针。
UINT	16 位无符号值。
WNDENUMPROC	指向一个回调函数 EnumWindowsProc 的 32 位指针。
WNDPROC	指向一个窗口过程的 32 位指针。
WORD	16 位无符号值。
WPARAM	作为一个参数传递给一个窗口过程或回调函数的 16 位带符号值。

第 2 章

消　息

Microsoft Windows 操作系统通过格式化的窗口消息与应用程序进行通信。这些消息发送到应用程序的窗口过程等候处理。

消息的返回值，或者包含有关这条消息成功的信息；或者包含应用程序所需要的其它数据。为了得到这个返回值，应用程序必须调用 SendMessage 函数将这条消息发送给一个窗口，这个函数一直到这条消息处理完后才返回。

如果应用程序不需要这条消息的返回值，则可以调用 PostMessage 函数来传递这条消息。这个函数将消息置于一个窗口的应用程序队列中，然后立即返回。如果一条消息没有返回值，应用程序可以用这两个函数中的任意一个来发送，除非在这条消息描述中另外指明。

一条消息由三部分组成：消息编号、字参数和长整数参数。消息编号由预定义的消息名称标识。每个消息名称由暗示这条消息的意义或来源的字母打头。字参数 *wParam* 和长整数参数 *lParam* 的值取决于消息编号。

lParam 参数常常包含一种类型以上的信息。例如，其高位字可能包含一个窗口句柄，而低位字可能包含一个整数值。实用宏 HIWORD 和 LOWORD 可以用来抽取 *lParam* 参数的高位字和低位字，它们可以用实用宏 HIBYTE 和 LOBYTE 来访问任意字节，也可以用计算方式来访问。

下面是消息编号的四个范围：

范　围	意　义
0 到 WM_USER-1	为 Windows 所用的消息而保留。
WM_USER 到 0x7FFF	为应用程序所用的整数消息。
0x8000 到 0xBFFF	为 Windows 所用的消息而保留。
0xC000 到 0xFFFF	为应用程序所用的串消息。

第一个范围（0 到 WM_USER-1）内的消息编号由 Windows 定义。在此范围内没有显示定义的值为 Windows 将来使用而保留。这一章描述了这个范围内的消息。

第二个范围（WM_USER 到 0x7FFF）内的消息编号，可以由应用程序定义并用来在一个私有窗口类内发送消息。例如 BUTTON，EDIT，LISTBOX 和 COMBOBOX 的预定义控制类可以利用这个范围内的值。此范围内的消息不应该发送给其它的应用程序，除非应用程序的设计支持交换消息并对这些消息编号赋予同样的意义。

第三个范围（0x8000 到 0xBFFF）内的消息编号，为 Windows 将来使用而保留。

第四个范围（0xC000 到 0xFFFF）内的消息编号，由应用程序在调用 RegisterWindowMessage 函数以获得一个字符串的消息编号时定义。所有注册相同字符串的应用程

序可以用相关的消息编号来互相交换消息。但是，实际的消息编号不是一个常量，不能认为在不同的 Windows 对话中是相同的。

2.1 窗口消息

这一部分描述窗口消息，这些消息按字母顺序排列。

BM_GETCHECK

2.x

```
BM_GETCHECK
wParam=0;  /*  不用，必须为零  */
lParam=0L;  /*  不用，必须为零  */
```

应用程序发送一条 BM_GETCHECK 消息，来检索一个按钮的检取状态。

参数 该消息没有参数。

返回值 从一个用 BS_AUTOCHECKBOX，BS_AUTORADIOBUTTON，BS_AUTO3STATE，BS_CHECKBOX，BS_RADIOBUTTON，或 BS_3STATE 风格创建的按钮返回值，可能是下列值中的一个：

值	意　义
0	按钮状态是未检取态。
1	按钮状态是检取态。
2	按钮状态是不确定的（只在按钮具有 BS_AUTO3STATE 或 BS_3STATE 风格时才应用）。

如果按钮具有其它风格，返回值是“0”。

实例 判断 ID_MYCHECKBOX 控制当前是否被检取：

```
int checked;
checked=(int) SendDlgItemMessage(hWndDlg, ID_MYCHECKBOX,
    BM_GETCHECK, 0, 0L);
```

参阅 BM_GETSTATE，BM_SETCHECK

BM_GETSTATE

2.x

```
BM_GETSTATE
wParam=0;  /*  不用，必须为零  */
lParam=0L;  /*  不用，必须为零  */
```

应用程序发送一条 BM_GETSTATE 消息，来检索一个按钮的状态。

参数 该消息没有参数。

返回值 返回值指明按钮的当前状态。可以利用下列屏蔽来抽取有关状态的信息：

屏 蔽	说 明
0x0003	指明检取状态(只用于无线按钮和检取框)。值为0表明按钮是未检取的,值为1表明按钮被检取。无线按钮中包含一个实点时表示被检取;检取框中包含一个 X 时表示被检取。值为2表明检取状态是不确定的(只用于三态检取框)。一个三态检取框的状态在它变灰时是不确定的。
0x0004	指明增亮状态。值为非零表明按钮被增亮。一个按钮在用户按下左鼠标按钮并保持时被增亮,当用户释放鼠标按钮时增亮消失。
0x0008	指明焦点状态。值为非零表明按钮拥有焦点。

实例 判断一个按钮当前是否拥有焦点。

```
#define  BFFOCUS  0x0008
DWORD  dwResult;
dwResult=SendDlgItemMessage(hdlg, ID_MYBUTTON, BM_GETSTATE, 0, 0L);
if  (dwResult & BFFOCUS)
/*  按钮拥有焦点  */
```

参阅 BM_GETCHECK, BM_SETSTATE

BM_SETCHECK

2.x

```
BM_SETCHECK
wParam=(WPARAM) fCheck;   /*  检取状态  */
lParam=0L;                /*  不用,必须为零  */
```

应用程序发送一条 BM_SETCHECK 消息来设置一个按钮的检取状态。

参数 *fCheck*

是 *wParam* 的值,指明检取状态。这个参数可以是下列值中的一个:

值	意 义
0	置按钮状态为未检取态。
1	置按钮状态为检取态。
2	置按钮状态为不确定态。这个值只在按钮具有 BS_3STATE 或 BS_AUTO3STATE 风格时才使用。

返回值 返回值总是零。

注释 BM_SETCHECK 消息对下压按钮没有影响。

实例 在一个无线按钮内放置一个实点:

```
SendDlgItemMessage (hdlg, ID_MYRADIOBUTTON, BM_SETCHECK, TRUE, 0L);
```

参阅 BM_GETCHECK, BM_GETSTATE, BM_SETSTATE

BM_SETSTATE

2.x

```
BM_SETSTATE
wParam=(WPARAM) fState;    /*  增亮状态  */
lParam=0L;                 /*  不用,必须为零  */
```

应用程序以发送一条 BM_SETSTATE 消息，来设置一个按钮的增亮状态。

参数 *fState*

是 *wParam* 的值，指明按钮是否要被增亮。值为非零则增亮按钮，值为零则撤销任何增亮。

返回值 返回值总是零。

注释 增亮影响一个按钮的外表，它对一个无线按钮或检取框的检取状态没有影响。一个按钮在用户按下左鼠标按钮并保持时自动增亮，当用户释放鼠标按钮时增亮消失。

实例 先增亮一个下压按钮，然后撤销这个增亮，用来模拟用户点取按钮时的直观效果：

```
SendDlgItemMessage (hdlg, ID_MYPUSHBUTTON, BM_SETSTATE, TRUE, 0L);
  /*
   * 执行一些动作;然后撤销增亮,
   * 而返回到它的正常状态.
   */
SendDlgItemMessage (hdlg, ID_MYPUSHBUTTON, BM_SETSTATE, FALSE, 0L);
```

参阅 BM_GETSTATE, BM_SETCHECK

BM_SETSTYLE

2.x

```
BM_SETSTYLE
wParam=(WPARAM) LOWORD (dwStyle);    /*  风格  */
lParam=MAKELPARAM (fRedraw, 0)       /*  重画标志  */
```

应用程序以发送一条 BM_SETSTYLE 消息，来改变一个按钮的风格。

参数 *dwStyle*

是 *wParam* 的值，指定按钮风格。按钮风格的说明请参看下面的注释部分。

fRedraw

是 *lParam* 低位字的值，指明按钮是否要被重画。值为 TRUE 则重画按钮，值为 FALSE 则不重画按钮。

返回值 返回值总是零。

注释 下面是按钮的风格：

值	意　义
BS_3STATE	创建一个和检取框类似的按钮，所不同的是它既可以被检取也可以变灰。变灰状态代表检取框已经被禁止。

（续表）

值	意　义
BS_AUTO3STATE	创建一个和三态检取框类似的按钮，所不同的是它在被用户选中之时改变自己的状态，它的状态在检取态、变灰态和常态之间循环。
BS_AUTOCHECKBOX	创建一个和检取框类似的按钮，所不同的是当被用户选中之时，在这个检取框内显示一个 X；在用户下次选择这个检取框时 X 消失（被清除）。
BS_AUTORADIOBUTTON	创建一个和无线按钮类似的按钮，所不同的是当被用户选中它时，该按钮自动地增亮并清除（撤销选择）同组内的其它按钮。
BS_CHECKBOX	创建一个将文本显示在其右边的小方块（除非这种风格和 BS_LEFTTEXT 风格相结合）。
BS_DEFPUSHBUTTON	创建一个具有深黑边框的按钮。用户可以通过按 ENTER 键来选择这个按钮。这种风格可用于支持用户快速选择最可能的选项（缺省选项）。
BS_GROUPBOX	创建一个对其它按钮进行分组的矩形。与这种风格有关的任何文本都显示在这个矩形的左上角处。
BS_LEFTTEXT	当和一个无线按钮或检取框风格相结合时，将文本置于这个无线按钮或检取框的左边。
BS_OWNERDRAW	创建一个自画式按钮。当创建这个按钮时拥有者窗口接收一个 WM_MEASUREITEM 消息；当这个按钮的直观外表改变时，拥有者窗口接收一条 WM_DRAWITEM 消息。BS_OWNERDRAW 风格不能与其它任何按钮风格相结合。
BS_PUSHBUTTON	创建一个下压按钮，该按钮在被用户选中之时将一条 WM_COMMAND 消息发送给拥有者窗口。
BS_RADIOBUTTON	创建一个将文本显示在其右边的小圆圈（除非这种风格与 BS_LEFTTEXT 风格相结合）。无线按钮通常适用于一组关联但互斥的选择。

应用程序不应该试图改变按钮的类型（例如，将一个无线按钮改变成一个检取框）。

实例　发送一条 BM_SETSTYLE 消息将一个按钮变成缺省下压按钮：

```
SendDlgItemMessage (hdlg, ID_MYPUSHBUTTON,BM_SETSTYLE,
    (WPARAM) BS_DEFPUSHBUTTON, TRUE);
```

CB_ADDSTRING　3.0

```
CB_ADDSTRING
wParam=0;                                /*  不用，必须为零   */
lParam=(LPARAM) (LPCSTR) lpsz;           /*  添加字符串的地址   */
```

应用程序发送一条 CB_ADDSTRING 消息，将一个字符串添加到一个组合框的

列表框中。如果该列表框不具备 CBS_SORT 风格，则将这个字符串添加到列表的尾部，否则将这个字符串插入到列表中，并将此列表重新排序。

参数 *lpsz*

是 *lParam* 的值，指向以 null 终结的添加字符串。如果组合框以自画式风格创建但不具备 CBS_HASSTRINGS 风格，则保存的是 *lpsz* 参数的值而不是它所指向的字符串。

返回值 返回值是列表框中字符串的索引，以零为基。如果发生错误则返回值为 CB_ERR；如果没有足够的空间，可用于保存新串，则返回值为 CB_ERRSPACE。

注释 如果自画式组合框以 CBS_SORT 风格创建，但不具备 CBS_HASSTRING 风格，则 WM_COMPAREITEM 消息多次被发送给组合框的拥有者，使新项能够正确地被置于列表框中。

要想将一个字符串插入到列表框的指定位置，可以利用 CB_INSERTSTRING 消息。

实例 将字符串"my string"插入到一个列表框中：

```
DWORD  dwIndex;
dwIndex=SendDlgItemMessage (hdlg, ID_MYCOMBOBOX,
    CB_ADDSTRING, 0, (LPARAM) ((LPCSTR) "my string"));
```

参阅 CB_INSERTSTRING, WM_COMPAREITEM

CB_DELETESTRING

3.0

```
CB_DELETESTRING
wParam=(WPARAM) index;   /*  删除项   */
lParam=0L;               /*  不用，必须为零   */
```

应用程序发送一条 CB_DELETESTRING 消息，来删除组合框的列表框中的一个字符串。

参数 *index*

是 *wParam* 的值。指定要删除字符串的索引，以零为基。

返回值 返回值是保留在列表中的字符串的个数。如果 *index* 参数指定的索引比列表中项的数目大，则返回值为 CB_ERR。

注释 如果组合框以自画式风格创建，但不具备 CBS_HASSTRING 风格，则发送一条 WM_DELETEITEM 消息给组合框的拥有者，使应用程序可以释放与该删除项有关的任何附加数据。

实例 删除一个组合框中的首串：

```
DWORD    dwRemaining;
dwRemaining=SendDlgItemMessage (hdlg,ID_MYCOMBOBOX,
    CB_DELETESTRING, 0, 0L);
```

参阅 WM_DELETEITEM