Microsoft® Windows

Software Development Kit

Quick Reference

Version 1.03

Microsoft Corporation

Document Number 050051054-103-I01-1086
Part Number 050-150-057

Contents

Introduction

The *Microsoft Windows Quick Reference* is a companion to the other manuals in the Microsoft Windows Software Development Kit. It is designed to provide brief, summarized information about Windows functions and messages and to provide page references to the *Microsoft Windows Programmer's Reference*, where more thorough descriptions of each function and message can be found.

Each function is listed alphabetically and is followed by a page reference to the *Microsoft Windows Programmer's Reference*.

Function parameters and return values have a lowercase prefix that indicates the general type of the parameter, followed by one or more words that describe the content of the parameter. The standard prefixes used in parameter and variable names are defined as follows:

Prefix	Meaning
c	Character (a one-byte value)
b	Boolean (a nonzero value means true, zero means false)
f	Bit flags packed into a 16-bit integer
n	Short (16-bit) integer
l	Long (32-bit) integer
w	Short (16-bit) unsigned integer
dw	Long (32-bit) unsigned integer
h	16-bit handle
p	Short (16-bit) pointer
lp	Long (32-bit) pointer
pt	*X, Y* coordinates packed into an unsigned 32-bit integer
rgb	An RGB color value packed into 32-bit integer

If no lowercase prefix is given, the parameter is a short integer whose name is descriptive.

A Windows Message Index lists each message alphabetically and gives a page reference to the *Microsoft Windows Programmer's Reference.* Following the Index, four alphabetic listings (Window Messages, Button-Control Messages, Edit-Control Messages, and List-Box Messages) describe the 16-bit and 32-bit parameters passed with each message. If a parameter is not used by the message, a hyphen (-) is given. A numeric list of Windows messages follows the alphabetic listing.

Windows Functions

AccessResource page 215

AccessResource(*hInstance,hResInfo*):*nFile*

Sets file pointer for read access to resource *hResInfo*. Returns DOS file handle.

AddAtom page 220

AddAtom(*lpString*):*wAtom*

Creates an atom for character string *lpString*.

AddFontResource page 207

AddFontResource(*lpFilename*):*nFonts*

Adds font resource in *lpFilename* to system font table.

AdjustWindowRect see Update

AdjustWindowRect(*lpRect,lStyle,bMenu*)

Converts client rectangle to a window rectangle.

AllocResource page 214

AllocResource(*hInstance,hResInfo,dwSize*):*hMem*

Allocates *dwSize* bytes of memory for resource *hResInfo*.

AnsiLower page 218

AnsiLower(*lpStr*):*cChar*

Converts character string *lpStr* to lowercase.

AnsiNext page 219

AnsiNext(*lpCurrentChar*):*lpNextChar*

Returns long pointer to next character in string *lpCurrentChar*.

AnsiPrev page 219

AnsiPrev(*lpStart,lpCurrentChar*):*lpPrevChar*

Returns long pointer to previous character in string *lpStart*. *lpCurrentChar* points to current character.

AnsiToOem page 217

AnsiToOem(*lpAnsiStr,lpOemStr*):*bTranslated*

Converts ANSI string to OEM character string.

AnsiUpper page 218

AnsiUpper(*lpStr*):*cChar*

Converts character string (or character if *lpString* high word is zero) to uppercase.

AnyPopup page 37

AnyPopup():*bVisible*

Indicates whether or not a popup style window is visible on the screen.

Arc page 114

Arc(*hDC,X1,Y1,X2,Y2,X3,Y3,X4,Y4*):*bDrawn*

Draws arc from *X3, Y3* to *X4, Y4*, using current pen and moving counterclockwise. Arc's center is at center of rectangle given by *X1, Y1* and *X2, Y2*.

BeginPaint page 73

BeginPaint(*hWnd,lpPaint*):*hDC*

Prepares window for painting, filling structure at *lpPaint* with painting data.

BitBlt page 116

BitBlt(*hDestDC,X,Y,nWidth,nHeight,hSrcDC,XSrc,YSrc,dwRop*):*bDrawn*

Moves bitmap from source device to destination device. Source origin is at *XSrc, YSrc*. *X, Y, nWidth, nHeight* give bitmap origin and dimensions on destination device. *dwRop* defines how source and destination bits are combined.

BringWindowToTop page 36

BringWindowToTop(*hWnd*)

Brings popup or child window to top of stack of overlapping windows.

BuildCommDCB page 228

BuildCommDCB(*lpDef,lpDCB*):*nResult*

Fills device control block *lpDCB* with control codes named by *lpDef.*

CallMsgFilter see Update

CallMsgFilter(*lpMsg,nCode*):*bResult*

Passes message and code to current message-filter function. Message-filter function is set using **SetWindowsHook**.

CallWindowProc page 25

CallWindowProc(*lpPrevWndFunc,hWnd,wMsg,wParam,lParam*):*lReply*

Passes message information to the function specified by *lpPrevWndFunc.*

Catch page 195

Catch(*lpCatchBuf*):*nThrowBack*

Copies current execution environment to buffer *lpCatchBuf.*

ChangeClipboardChain page 55

ChangeClipboardChain(*hWnd,hWndNext*):*bRemoved*

Removes *hWnd* from clipboard viewer chain, making *hWndNext* descendant of *hWnd*'s ancestor in the chain.

ChangeMenu page 62

ChangeMenu(*hMenu,wIDChangeItem,lpNewItem,wIDNewItem,*
wChange):*bChanged*

Appends, inserts, deletes, or modifies a menu item in *hMenu.*

CheckDlgButton page 45

CheckDlgButton(*hDlg,nIDButton,wCheck*)

Places or removes check next to button, or changes state of 3-state button.

CheckMenuItem page 65

CheckMenuItem(*hMenu,wIDCheckItem,wCheck*):*bOldCheck*

Places or removes checkmarks next to popup menu items in *hMenu.*

CheckRadioButton page 46

CheckRadioButton(*hDlg,nIDFirstButton,nIDLastButton,nIDCheckButton*)

Checks *nIDCheckButton* and unchecks all other radio buttons in the group from *nIDFirstButton* to *nIDLastButton.*

ChildWindowFromPoint page 96

ChildWindowFromPoint(*hWndParent,Point*):*hWndChild*

Determines which, if any, child window of *hWndParent* contains *Point.*

ClearCommBreak page 233

ClearCommBreak(*nCid*):*nResult*

Clears communication break state from communication device *nCid.*

ClientToScreen page 95

ClientToScreen(*hWnd,lpPoint*)

Converts client coordinates into equivalent screen coordinates in place.

ClipCursor page 91

ClipCursor(*lpRect*)

Restricts the mouse cursor to a given rectangle on the screen.

CloseClipboard page 47

CloseClipboard( ):*bClosed*

Closes the clipboard.

CloseComm page 226

CloseComm(*nCid*):*nResult*

Closes communication device *nCid* after transmitting current output buffer.

CloseMetaFile page 169

CloseMetaFile(*hDC***):***hMF*

Closes the metafile and creates a metafile handle.

CloseSound page 235

CloseSound( )

Closes play device after flushing voice queues and freeing buffers.

CloseWindow page 35

CloseWindow(*hWnd***):***nClosed*

Closes the specified window.

CombineRgn page 162

CombineRgn(*hDestRgn,hSrcRgn1,hSrcRgn2,nCombineMode***):***nRgnType*

Combines, using *nCombineMode*, two existing regions into a new region.

CopyMetaFile page 170

CopyMetaFile(*hSrcMetaFile,lpFilename***):***hMF*

Copies source metafile to *lpFilename* and returns the new metafile.

CopyRect page 97

CopyRect(*lpDestRect,lpSourceRect***)**

Makes a copy of an existing rectangle.

CountClipboardFormats page 53

CountClipboardFormats( ):*nCount*

Retrieves a count of the number of formats the clipboard can render.

CountVoiceNotes page 241

CountVoiceNotes(*nVoice***):***nNotes*

Returns number of notes in voice queue *nVoice*.

CreateBitmap page 131

CreateBitmap(*nWidth*,*nHeight*,*cPlanes*,*cBitCount*,*lpBits*):*hBitmap*

Creates a bitmap having the specified width, height, and bit pattern.

CreateBitmapIndirect page 132

CreateBitmapIndirect(*lpBitmap*):*hBitmap*

Creates a bitmap with the width, height, and bit pattern given by *lpBitmap*.

CreateBrushIndirect page 131

CreateBrushIndirect(*lpLogBrush*):*hBrush*

Creates a logical brush with the style, color, and pattern given by *lpLogBrush*.

CreateCaret page 93

CreateCaret(*hWnd*,*hBitmap*,*nWidth*,*nHeight*)

Creates caret for *hWnd* using *hBitmap*. If *hBitmap* is NULL, creates solid flashing black block *nWidth* by *nHeight* pixels; if *hBitmap* is 1, caret is gray.

CreateCompatibleBitmap page 132

CreateCompatibleBitmap(*hDC*,*nWidth*,*nHeight*):*hBitmap*

Creates a bitmap that is compatible with the device specified by *hDC*.

CreateCompatibleDC page 108

CreateCompatibleDC(*hDC*):*hMemDC*

Creates a memory display context compatible with the device specified by *hDC*.

CreateDC page 107

CreateDC(*lpDriverName*,*lpDeviceName*,*lpOutput*,*lpInitData*):*hDC*

Creates a display context for the specified device.

CreateDialog page 39

CreateDialog(*hInstance*,*lpTemplateName*,*hWndParent*,*lpDialogFunc*):*hDlg*

Creates a modeless dialog box.

CreateDiscardableBitmap see Update

CreateDiscardableBitmap(*hDC,X,Y*):*hBitmap*

Creates a discardable bitmap.

CreateEllipticRgn page 164

CreateEllipticRgn(*X1,Y1,X2,Y2*):*hRgn*

Creates an elliptical region whose bounding rectangle is defined by *X1*, *Y1*, *X2*, and *Y2*.

CreateEllipticRgnIndirect page 164

CreateEllipticRgnIndirect(*lpRect*):*hRgn*

Creates an elliptical region whose bounding rectangle is given by *lpRect*.

CreateFont page 134

CreateFont(*nHeight,nWidth,nEscapement,nOrientation,nWeight,cItalic,*
cUnderline,cStrikeOut,nCharSet,cOutputPrecision,
cClipPrecision,cQuality,cPitchAndFamily,lpFacename):*hFont*

Creates a logical font having the specified characteristics.

CreateFontIndirect page 137

CreateFontIndirect(*lpLogFont*):*hFont*

Creates a logical font with characteristics given by *lpLogFont*.

CreateHatchBrush page 130

CreateHatchBrush(*nIndex,rgbColor*):*hBrush*

Creates a logical brush having the specified hatched pattern and color.

CreateIC page 108

CreateIC(*lpDriverName,lpDeviceName,lpOutput,lpInitData*):*hIC*

Creates an information context for the specified device.

CreateMenu page 62

CreateMenu( ):*hMenu*

Creates an empty menu.

CreateMetaFile page 169

CreateMetaFile(*lpFilename*):*hDC*

Creates a metafile display context.

CreatePatternBrush page 130

CreatePatternBrush(*hBitmap*):*hBrush*

Creates a logical brush having the pattern specified by *hBitmap*.

CreatePen page 128

CreatePen(*nPenStyle,nWidth,rgbColor*):*hPen*

Creates a logical pen having the specified style, width, and color.

CreatePenIndirect page 129

CreatePenIndirect(*lpLogPen*):*hPen*

Creates a logical pen with the style, width, and color given by *lpLogPen*.

CreatePolygonRgn page 164

CreatePolygonRgn(*lpPoints,nCount,nPolyFillMode*):*hRgn*

Creates a polygonal region having *nCount* vertices as given by *lpPoints*.

CreateRectRgn page 163

CreateRectRgn(*X1, Y1,X2, Y2*):*hRgn*

Creates a rectangular region.

CreateRectRgnIndirect page 164

CreateRectRgnIndirect(*lpRect*):*hRgn*

Creates a rectangular region with the dimensions given by *lpRect*.

CreateSolidBrush page 129

CreateSolidBrush(*rgbColor*):*hBrush*

Creates a logical brush having the specified solid color.

CreateWindow page 26

CreateWindow(*lpClassName,lpWindowName,dwStyle,X,Y,nWidth,nHeight,hWndParent,hMenu,hInstance,lpParam*):*hWnd*

Creates tiled, popup, and child windows.

DefWindowProc page 21

DefWindowProc(*hWnd,wMsg,wParam,lParam*):*lReply*

Provides default processing for messages an application chooses not to process.

DeleteAtom page 220

DeleteAtom(*nAtom*):*nOldAtom*

Deletes an atom *nAtom* if its reference count is zero.

DeleteDC page 109

DeleteDC(*hDC*):*bDeleted*

Deletes the specified display context.

DeleteMetaFile page 170

DeleteMetaFile(*hMF*):*bFreed*

Deletes access to a metafile by freeing the associated system resources.

DeleteObject page 138

DeleteObject(*hObject*):*bDeleted*

Deletes the logical pen, brush, font, bitmap, or region by freeing all associated system storage.

DestroyCaret page 93

DestroyCaret( )

Destroys the current caret and frees any memory it occupied.

DestroyMenu page 62

DestroyMenu(*hMenu*):*bDestroyed*

Destroys the menu specified by *hMenu* and frees any memory it occupied.

DestroyWindow page 30

DestroyWindow(*hWnd*):*bDestroyed*

Sends a WM_DESTROY message to *hWnd* and frees any memory it occupied.

DeviceMode see Update

DeviceMode(*hWnd,hItem,lpString,lpString*):*lpString*

Displays a dialog box that prompts user to set printer modes.

DialogBox page 40

DialogBox(*hInstance,lpTemplateName,hWndParent,lpDialogFunc*):*nResult*

Creates a modal dialog box.

DispatchMessage page 17

DispatchMessage(*lpMsg*):*lResult*

Passes message to window function of window specified in **MSG** structure.

DlgDirList page 41

DlgDirList(*hDlg,lpPathSpec,nIDListBox,nIDStaticPath,wFiletype*):*nListed*

Fills *nIDListBox* with names of files matching path specification.

DlgDirSelect page 43

DlgDirSelect(*hDlg,lpString,nIDListBox*):*bDirectory*

Copies current selection from *nIDListBox* to *lpString*.

DPtoLP page 188

DPtoLP(*hDC,lpPoints,nCount*):*bConverted*

Converts into logical points the *nCount* device points given by *lpPoints*.

DrawIcon page 122

DrawIcon(*hDC,X,Y,hIcon*):*bDrawn*

Draws an icon with its upper left corner at *X*, *Y*.

DrawMenuBar page 65

DrawMenuBar(*hWnd*)

Redraws the menu bar.

DrawText page 119

DrawText(*hDC,lpString,nCount,lpRect,wFormat*)

Draws *nCount* characters of *lpString* in format specified by *wFormat*, using current text and background colors. Clips output to rectangle given by *lpRect*.

Ellipse page 113

Ellipse(*hDC,X1, Y1,X2, Y2*):*bDrawn*

Draws ellipse with center at the center of the given bounding rectangle. Draws border with current pen. Fills interior with current brush.

EmptyClipboard page 48

EmptyClipboard( ):*bEmptied*

Empties clipboard, frees data handles, and assigns clipboard ownership to the window that currently has the clipboard open.

EnableMenuItem page 66

EnableMenuItem(*hMenu,wIDEnableItem,wEnable*):*bEnabled*

Enables, disables, or grays a menu item, depending on *wEnable*.

EnableWindow page 60

EnableWindow(*hWnd,bEnable*):*bDone*

Enables and disables mouse and keyboard input to the specified window.

EndDialog page 41

EndDialog(*hDlg,nResult*)

Frees resources and destroys windows associated with a modal dialog box.

EndPaint page 74

EndPaint(*hWnd*,*lpPaint*)

Marks the end of window repainting; required after each **BeginPaint** call.

EnumChildWindows page 38

EnumChildWindows(*hWndParent*,*lpEnumFunc*,*lParam*):*bDone*

Enumerates the child style windows belonging to *hWndParent* by passing each child window handle and *lParam* to the *lpEnumFunc* function.

EnumClipboardFormats page 53

EnumClipboardFormats(*wFormat*):*wNextFormat*

Enumerates formats from list of available formats belonging to the clipboard.

EnumFonts page 180

EnumFonts(*hDC*,*lpFacename*,*lpFontFunc*,*lpData*):*nResult*

Enumerates fonts available on a given device, passing font information through *lpData* to *lpFontFunc* function.

EnumObjects page 181

EnumObjects(*hDC*,*nObjectType*,*lpObjectFunc*,*lpData*):*nResult*

Enumerates pens or brushes (depending on *nObjectType*) available on a device, passing object information through *lpData* to *lpObjectFunc* function.

EnumProps page 83

EnumProps(*hWnd*,*lpEnumFunc*):*nResult*

Passes each property of *hWnd*, in turn, to the *lpEnumFunc* function.

EnumWindows page 37

EnumWindows(*lpEnumFunc*,*lParam*):*bDone*

Enumerates windows on the screen by passing handle of each tiled, iconic, popup, and hidden popup window (in that order) to the *lpEnumFunc* function.

EqualRgn page 163

EqualRgn(*hSrcRgn1*,*hSrcRgn2*):*bEqual*

Checks the two given regions to determine if they are identical.

Escape page 171

Escape(*hDC*,*nEscape*,*nCount*,*lpInData*,*lpOutData*):*nResult*

Accesses device facilities not directly available through GDI.

Escape page 176

Escape(*hDC*,ABORTDOC,*nCount*,*lpInData*,*lpOutData*):*nResult*

Aborts the current job. *lpInData*, *lpOutData*, and *nCount* are not used.

Escape page 177

Escape(*hDC*,DRAFTMODE,*nCount*,*lpInData*,*lpOutData*):*nResult*

Turns draft mode off or on. *lpInData* points to 1 (on) or 0 (off). *nCount* is number of bytes at *lpInData*. *lpOutData* is not used.

Escape page 173

Escape(*hDC*,ENDDOC,*nCount*,*lpInData*,*lpOutData*):*nResult*

Ends print job started by STARTDOC. *nCount*, *lpInData*, *lpOutData* are not used.

Escape page 179

Escape(*hDC*,FLUSHOUTPUT,*nCount*,*lpInData*,*lpOutData*):*nResult*

Flushes output in device buffer; *lpInData*, *lpOutData*, and *nCount* are not used.

Escape page 178

Escape(*hDC*,GETCOLORTABLE,*nCount*,*lpInData*,*lpOutData*):*nResult*

Copies RGB color table entry to *lpOutData*. *lpInData* is color table index. *nCount* is not used.

Escape page 178

Escape(*hDC*,GETPHYSPAGESIZE,*nCount*,*lpInData*,*lpOutData*):*nResult*

Copies physical page size to **POINT** structure at *lpOutData*. *lpInData* and *nCount* are not used.

Escape page 178

Escape(*hDC*,GETPRINTINGOFFSET,*nCount*,*lpInData*,*lpOutData*):*nResult*

Copies printing offset to **POINT** structure at *lpOutData*. *lpInData* and *nCount* are not used.

Escape page 179

Escape(*hDC*,GETSCALINGFACTOR,*nCount*,*lpInData*,*lpOutData*):*nResult*

Copies scaling factors to **POINT** structure at *lpOutData*. *lpInData* and *nCount* are not used.

Escape page 174

Escape(*hDC*,NEWFRAME,*nCount*,*lpInData*,*lpOutData*):*nResult*

Ends writing to a page. *nCount*, *lpInData*, and *lpOutData* are not used.

Escape page 174

Escape(*hDC*,NEXTBAND,*nCount*,*lpInData*,*lpOutData*):*nResult*

Ends writing to a band. *lpOutData* gives rectangle to hold device coordinates of next band. *nCount* and *lpInData* are not used.

Escape page 179

Escape(*hDC*,QUERYESCSUPPORT,*nCount*,*lpInData*,*lpOutData*):*nResult*

Tests whether an escape is supported by device driver. *lpInData* points to the escape. *nCount* is the number of bytes at *lpInData*. *lpOutData* is not used.

Escape page 175

Escape(*hDC*,SETABORTPROC,*nCount*,*lpInData*,*lpOutData*):*nResult*

Sets abort function for print job. *lpInData*, *lpOutData*, and *nCount* are not used.

Escape page 178

Escape(*hDC*,SETCOLORTABLE,*nCount*,*lpInData*,*lpOutData*):*nResult*

Sets RGB color table entry. *lpInData* points to table index and color. *lpOutData* points to RGB color value to be set by device driver. *nCount* is not used.

Escape page 172

Escape(*hDC*,STARTDOC,*nCount*,*lpInData*,*lpOutData*):*nResult*

Starts print job, spooling NEWFRAME calls under same job until it reaches ENDDOC. *lpInData* is name of document; *nCount* is its length. *lpOutData* not used.

EscapeCommFunction page 234

EscapeCommFunction(*nCid*,*nFunc*):*nResult*

Executes escape function *nFunc* for communication device *nCid*.

ExcludeClipRect page 160

ExcludeClipRect(*hDC*,*X1*,*Y1*,*X2*,*Y2*):*nRgnType*

Creates new clipping region from existing clipping region less the given rectangle.

FatalExit page 225

FatalExit(*Code*):*Result*

Halts Windows and prompts through auxiliary port (AUX) for instructions on how to proceed.

FillRect page 126

FillRect(*hDC*,*lpRect*,*hBrush*):*nResult*

Fills given rectangle using the specified brush.

FillRgn page 125

FillRgn(*hDC*,*hRgn*,*hBrush*):*bFilled*

Fills given region with brush specified by *hBrush*.

FindAtom page 221

FindAtom(*lpString*):*wAtom*

Retrieves atom (if any) associated with character string *lpString*.

FindResource page 212

FindResource(*hInstance*,*lpName*,*lpType*):*hResInfo*

Locates resource *lpName* having *lpType* and returns handle for accessing and loading the resource.

FindWindow page 85

FindWindow(*lpClassName,lpWindowName*):*hWnd*

Returns the handle of the window having the given class and caption.

FlashWindow page 90

FlashWindow(*hWnd,bInvert*):*bInverted*

Flashes the given window once by inverting its active/inactive state.

FloodFill page 124

FloodFill(*hDC,X,Y,rgbColor*):*bFilled*

Fills area of the display surface with current brush, starting at *X, Y* and continuing in all directions to the boundaries with the given *rgbColor.*

FlushComm page 233

FlushComm(*nCid,nQueue*):*nResult*

Flushes characters from *nQueue* of communication device *nCid.*

FrameRect page 126

FrameRect(*hDC,lpRect,hBrush*):*nResult*

Draws border for the given rectangle using the specified brush.

FrameRgn page 125

FrameRgn(*hDC,hRgn,hBrush,nWidth,nHeight*):*bFramed*

Draws border for given region using *hBrush. nWidth* is width of vertical brush strokes. *nHeight* is height of horizontal strokes.

FreeLibrary page 194

FreeLibrary(*hLibModule*)

Removes library module *hLibModule* from memory if reference count is zero.

FreeProcInstance page 194

FreeProcInstance(*lpProc*)

Removes the function instance entry at address *lpProc.*

FreeResource page 215

FreeResource(*hResData*):*bFreed*

Removes resource *hResInfo* from memory if reference count is zero.

GetActiveWindow see Update

GetActiveWindow():*hWnd*

Returns handle to the active window.

GetAtomHandle see Update

GetAtomHandle(*wAtom*):*hMem*

Returns the handle (relative to the local heap) of the atom string.

GetAtomName page 221

GetAtomName(*wAtom,lpBuffer,nSize*):*nLength*

Copies character string (up to *nSize* characters) associated with *wAtom* to *lpBuffer*.

GetBitmapBits page 133

GetBitmapBits(*hBitmap,lCount,lpBits*):*lCopied*

Copies *lCount* bits of specified bitmap into buffer pointed to by *lpBits*.

GetBitmapDimension page 134

GetBitmapDimension(*hBitmap*):*ptDimensions*

Returns the width and height of the bitmap specified by *hBitmap*.

GetBkColor page 143

GetBkColor(*hDC*):*rgbColor*

Returns the current background color of the specified device.

GetBkMode page 144

GetBkMode(*hDC*):*nBkMode*

Returns the background mode of the specified device.

GetBrushOrg page 158

GetBrushOrg(*hDC*):*dwOrigin*

Retrieves the current brush origin for the given display context.

GetBValue page 244

GetBValue(*rgbColor*):*cBlue*

Retrieves the blue value of the given color.

GetCaretBlinkTime page 95

GetCaretBlinkTime( ):*wMSeconds*

Returns the current caret flash rate.

GetClassLong page 23

GetClassLong(*hWnd,nIndex*):*long*

Retrieves information at *nIndex* in the **WNDCLASS** structure.

GetClassName page 22

GetClassName(*hWnd,lpClassName,nMaxCount*):*nCopied*

Copies *hWnd*'s class name (up to *nMaxCount* characters) into *lpClassName.*

GetClassWord page 23

GetClassWord(*hWnd,nIndex*):*word*

Retrieves information at *nIndex* in the **WNDCLASS** structure.

GetClientRect page 85

GetClientRect(*hWnd,lpRect*)

Copies client coordinates of the window client area to *lpRect.*

GetClipboardData page 51

GetClipboardData(*wFormat*):*hClipData*

Retrieves data from the clipboard in the format given by *wFormat.*

GetClipboardFormatName page 53

GetClipboardFormatName(*wFormat,lpFormatName,nMaxCount***):***nCopied*

Copies *wFormat*'s format name (up to *nMaxCount* characters) into *lpFormatName.*

GetClipboardOwner page 48

GetClipboardOwner( ):*hWnd*

Retrieves the window handle of the current owner of the clipboard.

GetClipboardViewer page 54

GetClipboardViewer( ):*hWnd*

Retrieves the window handle of the first window in the clipboard viewer chain.

GetClipBox page 159

GetClipBox(*hDC,lpRect***):***nRgnType*

Copies dimensions of bounding rectangle of current clip boundary to *lpRect.*

GetCodeHandle page 193

GetCodeHandle(*lpFunc***):***hInstance*

Retrieves the handle of the code segment containing the given function.

GetCommError page 229

GetCommError(*nCid,lpStat***):***nError*

Fills buffer *lpStat* with communication status of device *nCid.* Returns error code, if any.

GetCommEventMask page 232

GetCommEventMask(*nCid,nEvtMask***):***wEvent*

Retrieves, then clears, event mask.

GetCommState page 229

GetCommState(*nCid,lpDCB***):***nResult*

Fills buffer *lpDCB* with the device control block of communication device *nCid.*

GetCurrentPosition page 111

GetCurrentPosition(*hDC***):***ptPos*

Retrieves the logical coordinates of the current position.

GetCurrentTask page 206

GetCurrentTask( ):*hTask*

Returns task handle of the current task.

GetCurrentTime page 15

GetCurrentTime( ):*lTime*

Returns the time elapsed since the system was booted to the current time.

GetCursorPos page 92

GetCursorPos(*lpPoint***)**

Stores mouse cursor position, in screen coordinates, in **POINT** structure.

GetDC page 70

GetDC(*hWnd***):***hDC*

Retrieves the display context for the client area of the specified window.

GetDeviceCaps page 183

GetDeviceCaps(*hDC,nIndex***):***nValue*

Retrieves the device-specific information specified by *nIndex*.

GetDlgItem page 43

GetDlgItem(*hDlg,nIDDlgItem***):***hCtl*

Retrieves the handle of a dialog item (control) from the given dialog box.

GetDlgItemInt page 44

GetDlgItemInt(*hDlg,nIDDlgItem,lpTranslated,bSigned*):*wValue*

Translates text of *nIDDlgItem* into integer value. Value at *lpTranslated* is zero if errors occur. *bSigned* is nonzero if minus sign might be present.

GetDlgItemText page 45

GetDlgItemText(*hDlg,nIDDlgItem,lpString,nMaxCount*):*nCopied*

Copies *nIDDlgItem*'s control text (up to *nMaxCount* characters) into *lpString.*

GetDoubleClickTime see Update

GetDoubleClickTime( ): *wClickTime*

Retrieves the current double-click time of the system mouse.

GetEnvironment page 186

GetEnvironment(*lpPortName,lpEnviron,nMaxCount*):*nCopied*

Copies to *lpEnviron* environment associated with device attached to given port.

GetFocus page 56

GetFocus( ):*hWnd*

Retrieves the handle of the window currently owning the input focus.

GetGVvalue page 244

GetGVValue(*rgbColor*): *cGreen*

Retrieves the green value of the given color.

GetInstanceData page 192

GetInstanceData(*hInstance,pData,nCount*):*nBytes*

Copies *nCount* bytes of data from offset *pData* in instance *hInstance* to same offset in current instance.

GetKeyState page 56

GetKeyState(*nVirtKey*):*nState*

Retrieves the state of the virtual key specified by *nVirtKey.*

GetMapMode page 151

GetMapMode(*hDC*):*nMapMode*

Retrieves the current mapping mode.

GetMenu page 62

GetMenu(*hWnd*):*hMenu*

Retrieves a handle to the menu of the specified window.

GetMenuString page 69

GetMenuString(*hMenu,wIDItem,lpString,nMaxCount,wFlag*):*nCopied*

Copies *wIDItem*'s menu label (up to *nMaxCount* characters) into *lpString*. *wFlag* is MF_BYPOSITION or MF_BYCOMMAND.

GetMessage page 10

GetMessage(*lpMsg,hWnd,wMsgFilterMin,wMsgFilterMax*):*bContinue*

Retrieves message in range *wMsgFilterMin* to *wMsgFilterMax*; stores at *lpMsg*.

GetMessagePos page 14

GetMessagePos( ):*dwPos*

Returns mouse position, in screen coordinates, at the time of the last message retrieved by **GetMessage**.

GetMessageTime page 14

GetMessageTime( ):*lTime*

Returns the message time for the last message retrieved by **GetMessage**.

GetMetaFile page 169

GetMetaFile(*lpFilename*):*hMF*

Creates a handle for the metafile named by *lpFilename*.

GetMetaFileBits page 170

GetMetaFileBits(*hMF*):*hMem*

Stores specified metafile as collection of bits in global memory block.

GetModuleFileName page 191

GetModuleFileName(*hModule,lpFilename,nSize*):*nLength*

Copies module filename (up to *nSize* characters) to *lpFilename.*

GetModuleHandle page 191

GetModuleHandle(*lpModuleName*):*hModule*

Returns module handle of module named by *lpModuleName.*

GetModuleUsage page 191

GetModuleUsage(*hModule*):*nCount*

Returns reference count of module *hModule.*

GetNearestColor page 187

GetNearestColor(*hDC,rgbColor*):*rgbSolidColor*

Returns the device color closest to *rgbColor.*

GetObject page 140

GetObject(*hObject,nCount,lpObject*):*nCopied*

Copies *nCount* bytes of logical data defining *hObject* to *lpObject.*

GetParent page 85

GetParent(*hWnd*):*hWndParent*

Retrieves the window handle of the specified window's parent (if any).

GetPixel page 123

GetPixel(*hDC,X,Y*):*rgbColor*

Retrieves the RGB color value of the pixel at the point specified by *X* and *Y.*

GetPolyFillMode page 148

GetPolyFillMode(*hDC*):*nPolyFillMode*

Retrieves the current polygon-filling mode.

GetProcAddress page 192

GetProcAddress(*hModule,lpProcName***):***lpAddress*

Returns address of the function named by *lpProcName* in module *hModule.*

GetProfileInt page 222

GetProfileInt(*lpSectionName,lpKeyName,nDefault***):***nKeyValue*

Returns integer value named by *lpKeyName* in section *lpSectionName* from the *win.ini* file. If name or section not found, *nDefault* is returned.

GetProfileString page 223

GetProfileString(*lpSectionName,lpKeyName,lpDefault,lpReturnedString,nSize***):***nLength*

Returns character string named by *lpKeyName* in section *lpSectionName* from the *win.ini* file. String is copied (up to *nSize* characters) to *lpReturnedString.* If name or section are not found, *lpDefault* is returned.

GetProp page 82

GetProp(*hWnd,lpString***):***hData*

Retrieves data handle associated with *lpString* from window property list.

GetRelAbs page 142

GetRelAbs(*hDC***):***nRelAbsMode*

Retrieves the relabs flag.

GetROP2 page 145

GetROP2(*hDC***):***nDrawMode*

Retrieves the current drawing mode.

GetRValue page 244

GetRValue(*rgbColor***):** *cRed*

Retrieves the red value of the given color.

GetScrollPos page 79

GetScrollPos(*hWnd,nBar***):***nPos*

Retrieves current position of scroll bar elevator identified by *hWnd* and *nBar*.

GetScrollRange page 81

GetScrollRange(*hWnd,nBar,lpMinPos,lpMaxPos***)**

Copies minimum and maximum scroll bar positions for given scroll bar to *lpMinPos* and *lpMaxPos*.

GetStockObject page 127

GetStockObject(*nIndex***):***hObject*

Retrieves a handle to a predefined stock pen, brush, or font.

GetStretchBltMode page 147

GetStretchBltMode(*hDC***):***nStretchMode*

Retrieves the current stretching mode.

GetSubMenu page 68

GetSubMenu(*hMenu,nPos***):***hPopupMenu*

Retrieves the menu handle of the popup menu at the given position in *hMenu*.

GetSysColor page 101

GetSysColor(*nIndex***):***rgbColor*

Retrieves the system color identified by *nIndex*.

GetSysModalWindow page 86

GetSysModalWindow():*hWnd*

Returns the handle of a system-modal window, if one is present.

GetSystemMenu page 68

GetSystemMenu(*hWnd,bRevert***):***hSysMenu*

Allows access to the System menu for copying and modification. *bRevert* is nonzero to restore the original System menu.

GetSystemMetrics page 100

GetSystemMetrics(*nIndex*):*nValue*

Retrieves information about the system metrics identified by *nIndex*.

GetTempDrive page 248

GetTempDrive(*cDriveLetter*):*cOptDriveLetter*

Returns letter for the optimal drive for a temporary file. *cDriveLetter* is a proposed drive.

GetTempFileName page 248

GetTempFileName(*cDriveLetter,lpPrefixString,wUnique,*
lpTempFileName):*wUniqueNumber*

Creates a temporary filename.

GetTextCharacterExtra page 167

GetTextCharacterExtra(*hDC*):*nCharExtra*

Retrieves the current intercharacter spacing.

GetTextColor page 144

GetTextColor(*hDC*):*rgbColor*

Retrieves the current text color.

GetTextExtent page 166

GetTextExtent(*hDC,lpString,nCount*):*dwTextExtents*

Uses current font to compute width and height of text line given by *lpString*.

GetTextFace page 182

GetTextFace(*hDC,nCount,lpFacename*):*nCopied*

Copies the current font's facename (up to *nCount* characters) into *lpFacename*.

GetTextMetrics page 183

GetTextMetrics(*hDC,lpMetrics*):*bRetrieved*

Fills buffer given by *lpMetrics* with metrics for currently selected font.

GetThresholdEvent page 241

GetThresholdEvent( ):*lpInt*

Returns long pointer to a threshold flag. The flag is set if any voice queue is below threshold (i.e., below a given number of notes).

GetThresholdStatus page 241

GetThresholdStatus( ):*fStatus*

Returns a bit mask containing the threshold event status. If a bit is set, the given voice queue is below threshold.

GetUpdateRect page 75

GetUpdateRect(*hWnd,lpRect,bErase***):***bUpdate*

Copies dimensions of bounding rectangle of window region that needs updating to *lpRect*. *bErase* is nonzero if background needs erasing. *bUpdate* is zero if window is up-to-date.

GetVersion page 195

GetVersion( ):*wVersion*

Returns the current version of Windows.

GetViewportExt page 158

GetViewportExt(*hDC***):***ptExtents*

Retrieves the *x*- and *y*-extents of the display context's viewport.

GetViewportOrg page 154

GetViewportOrg(*hDC***):***ptOrigin*

Retrieves *x*- and *y*-coordinates of the origin of the display context's viewport.

GetWindowDC page 72

GetWindowDC(*hWnd***):***hDC*

Retrieves display context for entire window, including caption bar, menus, scroll bars.

GetWindowExt page 153

GetWindowExt(*hDC*):*ptExtents*

Retrieves *x*- and *y*-extents of the display context's window.

GetWindowLong page 31

GetWindowLong(*hWnd,nIndex*):*long*

Retrieves information identified by *nIndex* about the given window.

GetWindowOrg page 151

GetWindowOrg(*hDC*):*ptOrigin*

Retrieves *x*- and *y*-coordinates of the origin of the display context's window.

GetWindowRect page 86

GetWindowRect(*hWnd,lpRect*)

Copies dimensions, in screen coordinates, of entire window (including caption bar, border, menus, and scroll bars) to *lpRect.*

GetWindowText page 84

GetWindowText(*hWnd,lpString,nMaxCount*):*nCopied*

Copies *hWnd*'s window caption (up to *nMaxCount* characters) into *lpString.*

GetWindowTextLength page 84

GetWindowTextLength(*hWnd*):*nLength*

Returns the length of the given window's caption or text.

GetWindowWord page 30

GetWindowWord(*hWnd,nIndex*):*word*

Retrieves information identified by *nIndex* about the given window.

GlobalAlloc page 196

GlobalAlloc(*wFlags,dwBytes*):*hMem*

Allocates *dwBytes* of memory from the global heap. Memory type (e.g., fixed or moveable) is set by *wFlags.*

GlobalCompact page 197

GlobalCompact(*dwMinFree*):*dwLargest*

Compacts global memory to generate *dwMinFree* free bytes.

GlobalDiscard page 197

GlobalDiscard(*hMem*):*hOldMem*

Discards global memory block *hMem* if reference count is zero.

GlobalFlags page 200

GlobalFlags(*hMem*):*wFlags*

Returns memory type of global memory block *hMem*.

GlobalFree page 197

GlobalFree(*hMem*):*hOldMem*

Removes global memory block *hMem* from memory if reference count is zero.

GlobalHandle see Update

GlobalHandle(*wMem*): *dwMem*

Retrieves the handle of the global memory object whose segment address is *wMem*.

GlobalLock page 198

GlobalLock(*hMem*):*lpAddress*

Returns address of global memory block *hMem*, locks block in memory, and increases the reference count by one.

GlobalReAlloc page 198

GlobalReAlloc(*hMem*,*dwBytes*,*wFlags*):*hNewMem*

Reallocates the global memory block *hMem* to *dwBytes* and memory type *wFlags*.

GlobalSize page 199

GlobalSize(*hMem*):*dwBytes*

Returns the size, in bytes, of global memory block *hMem*.

GlobalUnlock page 200

GlobalUnlock(*hMem***):***bResult*

Unlocks global memory block *hMem* and decreases the reference count by one.

GrayString page 121

GrayString(*hDC,hBrush,lpOutputFunc,lpData,nCount,X,Y,nWidth,nHeight***):***bDrawn*

Writes *nCount* characters of string at *X, Y,* using *lpOutputFunc* (or **TextOut** if NULL). Grays text using *hBrush. lpData* specifies output string (if *lpOutputFunc* is NULL) or data are passed to output function. *nWidth* and *nHeight* give dimensions of enclosing rectangle (if zero, dimensions are calculated).

HIBYTE page 242

HIBYTE(*nInteger***):***cHighByte*

Returns the high-order byte of *nInteger.*

HideCaret page 94

HideCaret(*hWnd***)**

Removes system caret from the given window.

HiliteMenuItem page 67

HiliteMenuItem(*hWnd,hMenu,wIDHiliteItem,wHilite***):***bHilited*

Highlights or removes the highlighting from a top-level (menu bar) menu item.

HIWORD page 243

HIWORD(*lInteger***):***wHighWord*

Returns the high-order word of *lInteger.*

InflateRect page 98

InflateRect(*lpRect,X,Y***):***nResult*

Expands or shrinks the rectangle specified by *lpRect* by *X* units on the left and right ends of the rectangle and *Y* units on the top and bottom.

InitAtomTable page 219

InitAtomTable(*nSize***):***bResult*

Initializes atom hash table and sets it to *nSize* atoms.

InSendMessage see Update

InSendMessage():*bInSend*

Returns TRUE if window function is processing a message sent with **SendMessage**.

IntersectClipRect page 159

IntersectClipRect(*hDC,X1,Y1,X2,Y2***):***nRgnType*

Forms new clipping region from intersection of current clipping region and given rectangle.

IntersectRect page 98

IntersectRect(*lpDestRect,lpSrc1Rect,lpSrc2Rect***):***nIntersection*

Finds the intersection of two rectangles and copies it to *lpDestRect*.

InvalidateRect page 75

InvalidateRect(*hWnd,lpRect,bErase***)**

Marks for repainting the rectangle specified by *lpRect* (in client coordinates). The rectangle is erased if *bErase* is nonzero.

InvalidateRgn page 76

InvalidateRgn(*hWnd,hRgn,bErase***)**

Marks *hRgn* for repainting. The region is erased if *bErase* is nonzero.

InvertRect page 127

InvertRect(*hDC,lpRect***):***nResult*

Inverts the display bits of the specified rectangle.

InvertRgn page 126

InvertRgn(*hDC,hRgn***):***bInverted*

Inverts the colors in the region specified by *hRgn*.

IsChild see Update

IsChild(*hParentWnd,hWnd*):*bChild*

Returns TRUE if given window is a child of *hParentWnd*.

IsClipboardFormatAvailable see Update

IsClipboardFormatAvailable(*wFormat*):*bAvailable*

Returns TRUE if data in given format is available.

IsDialogMessage page 40

IsDialogMessage(*hDlg,lpMsg*):*bUsed*

Determines whether *lpMsg* is intended for the given modeless dialog box. If so, the message is processed and *bUsed* is nonzero.

IsDlgButtonChecked page 45

IsDlgButtonChecked(*hDlg,nIDButton*):*wCheck*

Tests whether *nIDButton* is checked. For a 3-state button, returns 2 for grayed, 1 for checked, zero for neither.

IsIconic page 37

IsIconic(*hWnd*):*bIconic*

Specifies whether or not a window is open or closed (iconic).

IsRectEmpty page 99

IsRectEmpty(*lpRect*):*bEmpty*

Determines whether or not the specified rectangle is empty.

IsWindow page 29

IsWindow(*hWnd*):*bExists*

Determines whether or not *hWnd* is a valid, existing window.

IsWindowEnabled page 61

IsWindowEnabled(*hWnd*):*bEnabled*

Specifies whether or not *hWnd* is enabled for mouse and keyboard input.

IsWindowVisible page 36

IsWindowVisible(*hWnd*):*bVisible*

Determines whether or not the given window is visible on the screen.

KillTimer page 60

KillTimer(*hWnd,nIDEvent*):*bKilled*

Kills the timer event identified by *hWnd* and *nIDEvent*.

LineDDA page 124

LineDDA(*X1, Y1, X2, Y2, lpLineFunc, lpData*)

Computes successive points in line starting at *X1, Y1* and ending at *X2, Y2*, passing each point and *lpData* parameter to *lpLineFunc* function.

LineTo page 111

LineTo(*hDC,X, Y*):*bDrawn*

Draws line with current pen from the current position up to, but not including, the point *X, Y*.

LoadAccelerators page 212

LoadAccelerators(*hInstance,lpTableName*):*hRes*

Loads accelerator table named by *lpTableName*.

LoadBitmap page 209

LoadBitmap(*hInstance,lpBitmapName*):*hBitmap*

Loads bitmap resource named by *lpBitmapName*.

LoadCursor page 209

LoadCursor(*hInstance,lpCursorName*):*hCursor*

Loads cursor resource named by *lpCursorName*.

LoadIcon page 210

LoadIcon(*hInstance,lpIconName*):*hIcon*

Loads icon resource named by *lpIconName*.

LoadLibrary page 194

LoadLibrary(*lpLibFileName*):*hLibModule*

Loads the library module named by *lpLibFilename.*

LoadMenu page 211

LoadMenu(*hInstance,lpMenuName*):*hMenu*

Loads menu resource named by *lpMenuName.*

LoadResource page 214

LoadResource(*hInstance,hResInfo*):*hResData*

Loads the resource *hResInfo* and returns a handle to the resource.

LoadString page 212

LoadString(*hInstance,wID,lpBuffer,nBufferMax*):*nSize*

Loads string resource *wID* into the buffer *lpBuffer.* Up to *nBufferMax* characters are copied.

LOBYTE page 242

LOBYTE(*nInteger*):*cLowByte*

Returns the low-order byte of *nInteger.*

LocalAlloc page 200

LocalAlloc(*wFlags,wBytes*):*hMem*

Allocates *wBytes* of memory from the local heap. Memory type (e.g., fixed or moveable) is set by *wFlags.*

LocalCompact page 201

LocalCompact(*wMinFree*):*wLargest*

Compacts local memory to generate *wMinFree* free bytes.

LocalDiscard page 202

LocalDiscard(*hMem*):*hOldMem*

Discards local memory block *hMem* if reference count is zero.

LocalFlags page 206

LocalFlags(*hMem*):*wFlags*

Returns memory type of local memory block *hMem*.

LocalFree page 202

LocalFree(*hMem*):*hOldMem*

Frees local memory block *hMem* from memory if reference count is zero.

LocalFreeze page 202

LocalFreeze(*Dummy*)

Prevents compaction of the local heap.

LocalHandle see Update

LocalHandle(*wMem*):*hMem*

Retrieves the handle of the local memory object whose address is *wMem*.

LocalHandleDelta page 205

LocalHandleDelta(*nNewDelta*):*nCurrentDelta*

Sets the entry count for each new handle table created in the local heap.

LocalInit see Update

LocalInit(*wValue,pString,pString*):*bResult*

Initializes the local heap.

LocalLock page 202

LocalLock(*hMem*):*pAddress*

Returns the address of local memory block *hMem*, locks the block in memory, and increases the reference count by one.

LocalMelt page 203

LocalMelt(*Dummy*)

Permits compaction of the local heap.

LocalNotify see Update

LocalNotify(*lpFunc***):***lpPrevFunc*

Sets the callback function for handling notification messages from local memory.

LocalReAlloc page 203

LocalReAlloc(*hMem,wBytes,wFlags***):***hNewMem*

Reallocates the local memory block *hMem* to *wBytes* and memory type *wFlags*.

LocalSize page 204

LocalSize(*hMem***):***wBytes*

Returns the size, in bytes, of local memory block *hMem*.

LocalUnlock page 205

LocalUnlock(*hMem***):***bResult*

Unlocks local memory block *hMem* and decreases the reference count by one.

LockData page 205

LockData(*Dummy***):***hMem*

Locks the data segment in memory.

LockResource page 214

LockResource(*hResInfo***):***lpResInfo*

Returns the memory address of the resource *hResInfo*, locks the resource in memory, and increases the reference count by one.

LockSegment Function see Update

LockSegment(*wSegment***):***hSegment*

Locks the segment whose segment address is *wSegment*.

LOWORD page 243

LOWORD(*lInteger***):***wLowWord*

Returns the low-order word of *lInteger*.

LPtoDP page 188

LPtoDP(*hDC,lpPoints,nCount*):*bConverted*

Converts logical points into device points.

MAKEINTATOM page 221

MAKEINTATOM(*wInteger*):*nAtom*

Casts an integer for use as an argument to **AddAtom**.

MAKEINTRESOURCE page 243

MAKEINTRESOURCE(*nInteger*):*lpIntegerID*

Casts an integer for use as an argument in **AddAtom**.

MAKELONG page 243

MAKELONG(*nLowWord,nHighWord*):*dwInteger*

Creates an unsigned long integer.

MAKEPOINT page 244

MAKEPOINT(*lValue*):*ptPoint*

Converts a long value into a **POINT** structure.

MakeProcInstance page 193

MakeProcInstance(*lpProc,hInstance*):*lpAddress*

Returns a function instance address for function *lpProc*. Calls to the instance address ensure that the function uses the data segment of instance *hInstance*.

MapDialogRect page 47

MapDialogRect(*hDlg,lpRect*)

Converts the dialog box coordinates given in *lpRect* to client coordinates.

max page 245

max(*A,B*):*nMaximum*

Returns the maximum value of *A* and *B*.

MessageBeep page 89

MessageBeep(*wType*):*bBeep*

Generates a beep at the system speaker when a message box is displayed.

MessageBox page 87

MessageBox(*hWndParent,lpText,lpCaption,wType*):*nMenuItem*

Creates window with given *lpText* and *lpCaption* containing the predefined icons and push buttons defined by *wType*.

min page 244

min(*A,B*):*nMinimum*

Returns the minimum value of *A* and *B*.

MoveTo page 110

MoveTo(*hDC,X,Y*):*ptPrevPos*

Moves the current position to the point specified by *X* and *Y*.

MoveWindow page 35

MoveWindow(*hWnd,X,Y,nWidth,nHeight,bRepaint*)

Causes WM_SIZE message to be sent to *hWnd*. *X*, *Y*, *nWidth*, and *nHeight* give the new size of the window.

OemToAnsi page 218

OemToAnsi(*lpOemStr,lpAnsiStr*):*bTranslated*

Converts the OEM character string to an ANSI string.

OffsetClipRgn page 160

OffsetClipRgn(*hDC,X,Y*):*nRgnType*

Moves clipping region *X* units along the *x*-axis and *Y* units along the *y*-axis.

OffsetRect page 99

OffsetRect(*lpRect,X,Y*):*nResult*

Moves given rectangle *X* units along the *x*-axis and *Y* units along the *y*-axis.

OffsetRgn page 163

OffsetRgn(*hRgn*,*X*,*Y*):*nRgnType*

Moves the given region *X* units along the *x*-axis and *Y* units along the *y*-axis.

OpenClipboard page 47

OpenClipboard(*hWnd*):*bOpened*

Opens clipboard; prevents other applications from modifying its contents.

OpenComm page 225

OpenComm(*lpComName*,*wInQueue*,*wOutQueue*):*nCid*

Opens communication device named by *lpCommName*. Transmit-queue and receive-queue sizes are set by *wInQueue* and *wOutQueue*.

OpenFile page 245

OpenFile(*lpFileName*,*lpReOpenBuff*,*wStyle*):*nFile*

Creates, opens, reopens, or deletes file named by *lpFileName*.

OpenIcon page 34

OpenIcon(*hWnd*):*bOpened*

Opens the specified window.

OpenSound page 235

OpenSound( ):*nVoices*

Opens the play device for exclusive use.

PaintRgn page 126

PaintRgn(*hDC*,*hRgn*):*bFilled*

Fills the region specified by *hRgn* with the currently selected brush.

PatBlt page 115

PatBlt(*hDC*,*X*,*Y*,*nWidth*,*nHeight*,*dwRop*):*bDrawn*

Creates a bit pattern on the specified device, using *dwRop* to combine the current brush with the pattern already on the device.

PeekMessage page 12

PeekMessage(*lpMsg,hWnd,wMsgFilterMin,wMsgFilterMax,bRemoveMsg*):*bPresent*

Checks application queue and places message (if any) at *lpMsg.*

Pie page 115

Pie(*hDC,X1,Y1,X2,Y2,X3,Y3,X4,Y4*):*bDrawn*

Draws arc starting at *X3, Y3* and ending at *X4, Y4* and connects center and two endpoints, using current pen. Moves counterclockwise. Fills with current brush. Arc's center is center of bounding rectangle given by *X1, Y1, X2, Y2.*

PlayMetaFile page 170

PlayMetaFile(*hDC,hMF*):*bPlayed*

Plays the contents of the specified metafile on the given device context.

Polygon page 113

Polygon(*hDC,lpPoints,nCount*):*bDrawn*

Draws a polygon by connecting the *nCount* vertices given by *lpPoints.*

Polyline page 111

Polyline(*hDC,lpPoints,nCount*):*bDrawn*

Draws a set of line segments, connecting the *nCount* points given by *lpPoints.*

PostAppMessage page 19

PostAppMessage(*hTask,wMsg,wParam,lParam*):*bPosted*

Posts message to application; returns without waiting for processing.

PostMessage page 18

PostMessage(*hWnd,wMsg,wParam,lParam*):*bPosted*

Places message in application queue; returns without waiting for processing.

PostQuitMessage page 10

PostQuitMessage(*nExitCode*)

Posts a WM_QUIT message to the application and returns immediately.

PtInRect page 99

PtInRect(*lpRect,Point*):*bInRect*

Indicates whether or not a specified point lies within a given rectangle.

PtInRegion page 165

PtInRegion(*hRgn,X,Y*):*bSuccess*

Tests if *X, Y* is within the given region.

PtVisible page 161

PtVisible(*hDC,X,Y*):*bVisible*

Tests if *X, Y* is within the clipping region of the given display context.

ReadComm page 227

ReadComm(*nCid,lpBuf,nSize*):*nBytes*

Reads up to *nSize* bytes from the communication device *nCid* into buffer *lpBuf.*

Rectangle page 112

Rectangle(*hDC,X1,Y1,X2,Y2*):*bDrawn*

Draws rectangle, using current pen for border and current brush for filling.

RectVisible page 161

RectVisible(*hDC,lpRect*):*bVisible*

Determines if any part of given rectangle lies within clipping region.

RegisterClass page 22

RegisterClass(*lpWndClass*):*bRegistered*

Registers a window class.

RegisterClipboardFormat page 52

RegisterClipboardFormat(*lpFormatName*):*wFormat*

Registers a new clipboard format whose name is pointed to by *lpFormatName.*

RegisterWindowMessage page 19

RegisterWindowMessage(*lpString*):*wMsg*

Defines a new window message that is guaranteed to be unique.

ReleaseCapture page 59

ReleaseCapture( )

Releases mouse input and restores normal input processing.

ReleaseDC page 73

ReleaseDC(*hWnd,hDC*):*nReleased*

Releases a display context when an application is finished drawing in it.

RemoveFontResource page 208

RemoveFontResource(*lpFilename*):*bSuccess*

Removes from the font table the font resource named by *lpFilename.*

RemoveProp page 82

RemoveProp(*hWnd,lpString*):*hData*

Removes *lpString* from property list; retrieves corresponding data handle

ReplyMessage page 19

ReplyMessage(*lReply*)

Replies to message without returning control to the **SendMessage** caller.

RestoreDC page 110

RestoreDC(*hDC,nSavedDC*):*bRestored*

Restores display context given by *hDC* to previous state given by *nSavedDC.*

RGB see Update

RGB(r,g,b):*DWORD*

Creates an RGB color value from individual red, green, and blue values.

RoundRect page 112

RoundRect(*hDC,X1,Y1,X2,Y2,X3,Y3*):*bDrawn*

Draws rounded rectangle, using current pen for border, current brush for filling.

SaveDC page 109

SaveDC(*hDC*):*nSavedDC*

Saves the current state of the display context *hDC*.

ScreenToClient page 96

ScreenToClient(*hWnd,lpPoint*)

Converts the screen coordinates at *lpPoint* to client coordinates.

ScrollWindow page 77

ScrollWindow(*hWnd,XAmount,YAmount,lpRect,lpClipRect*)

Moves contents of client area *XAmount* along screen's *x*-axis and *YAmount* units along *y*-axis (right for positive *XAmount*; down for positive *YAmount*).

SelectClipRgn page 139

SelectClipRgn(*hDC,hRgn*):*nRgnType*

Selects given region as current clipping region for the specified display context.

SelectObject page 138

SelectObject(*hDC,hObject*):*hOldObject*

Selects *hObject* as current object, replacing previous object of same type.

SendDlgItemMessage page 46

SendDlgItemMessage(*hDlg,nIDDlgItem,wMsg,wParam,lParam*):*lResult*

Sends a message to *nIDDlgItem* within the dialog box specified by *hDlg*.

SendMessage page 17

SendMessage(*hWnd,wMsg,wParam,lParam*):*lReply*

Sends a message to a window or windows.

SetActiveWindow page 36

SetActiveWindow(*hWnd*):*hWndPrev*

Makes a tiled or popup style window the active window.

SetBitmapBits page 133

SetBitmapBits(*hBitmap*,*dwCount*,*lpBits*):*bCopied*

Sets bitmap bits to values given at *lpBits*. *dwCount* is byte count at *lpBits*.

SetBitmapDimension page 133

SetBitmapDimension(*hBitmap*,*X*,*Y*):*ptOldDimensions*

Associates a width and height, in 0.1 millimeter units, with a bitmap.

SetBkColor page 142

SetBkColor(*hDC*,*rgbColor*):*rgbOldColor*

Sets current background color to the device color closest to *rgbColor*.

SetBkMode page 143

SetBkMode(*hDC*,*nBkMode*):*nOldBkMode*

Sets the background mode used with text, hatched brushes, and line styles.

SetBrushOrg page 158

SetBrushOrg(*hDC*,*X*,*Y*):*dwOldOrigin*

Sets the origin of all brushes selected into the given display context.

SetCapture page 58

SetCapture(*hWnd*):*hWndPrev*

Causes mouse input to be sent to *hWnd*, regardless of mouse cursor position.

SetCaretBlinkTime page 95

SetCaretBlinkTime(*wMSeconds*)

Establishes the caret flash rate.

SetCaretPos page 94

SetCaretPos(*X*, *Y*)

Moves the caret to the position specified by *X* and *Y*.

SetClassLong page 24

SetClassLong(*hWnd*,*nIndex*,*lNewLong*):*lOldLong*

Replaces long value at *nIndex* in the **WNDCLASS** structure.

SetClassWord page 24

SetClassWord(*hWnd*,*nIndex*,*wNewWord*):*wOldWord*

Replaces word at the given *nIndex* in the **WNDCLASS** structure.

SetClipboardData page 48

SetClipboardData(*wFormat*,*hMem*):*hClipData*

Copies *hMem*, a handle for data having *wFormat* format, into the clipboard.

SetClipboardViewer page 54

SetClipboardViewer(*hWnd*):*hWndNext*

Adds *hWnd* to clipboard viewer chain. *hWndNext* is next window in chain.

SetCommBreak page 233

SetCommBreak(*nCid*):*nResult*

Sets a break state on communication device *nCid* and suspends character transmission.

SetCommEventMask page 231

SetCommEventMask(*nCid*,*nEvtMask*):*lpEvent*

Sets the event mask of the communication device *nCid*.

SetCommState page 229

SetCommState(*lpDCB*):*nResult*

Sets a communication device to the state specified by the device control block *lpDCB*. The device to be set is identified by the ID field of the control block.

SetCursor page 91

SetCursor(*hCursor*):*hOldCursor*

Sets cursor shape to *hCursor*; removes cursor from screen if *hCursor* is NULL.

SetCursorPos page 91

SetCursorPos(*X*, *Y*)

Sets position of mouse cursor to screen coordinates given by *X* and *Y*.

SetDlgItemInt page 43

SetDlgItemInt(*hDlg*,*nIDDlgItem*,*wValue*,*bSigned*)

Sets text of *nIDDlgItem* to string representing an integer.

SetDlgItemText page 44

SetDlgItemText(*hDlg*,*nIDDlgItem*,*lpString*)

Sets caption or text of *nIDDlgItem* to *lpString*.

SetEnvironment page 186

SetEnvironment(*lpPortName*,*lpEnviron*,*nCount*):*nCopied*

Copies data at *lpEnviron* to environment associated with device attached to given port.

SetFocus page 56

SetFocus(*hWnd*):*hWndPrev*

Assigns the input focus to the window specified by *hWnd*.

SetMapMode page 148

SetMapMode(*hDC*,*nMapMode*):*nOldMapMode*

Sets the mapping mode of the specified display context.

SetMenu page 61

SetMenu(*hWnd*,*hMenu*):*bSet*

Sets window menu to *hMenu*. Removes menu if *hMenu* is NULL.

SetMetaFileBits page 171

SetMetaFileBits(*hMem*):*hMF*

Creates memory metafile from data in the given global memory block.

SetPixel page 123

SetPixel(*hDC,X,Y,rgbColor*):*rgbActualColor*

Sets pixel at *X, Y* to the device color closest to *rgbColor*.

SetPolyFillMode page 147

SetPolyFillMode(*hDC,nPolyFillMode*):*nOldPolyFillMode*

Sets the polygon-filling mode for the specified display context.

SetPriority page 207

SetPriority(*hTask,nChangeAmount*):*nNew*

Sets the task priority of the task *hTask*, and returns new priority.

SetProp page 82

SetProp(*hWnd,lpString,hData*):*bSet*

Copies string and data handle to property list of *hWnd*.

SetRect page 97

SetRect(*lpRect,X1,Y1,X2,Y2*):*nResult*

Fills **RECT** structure at *lpRect* with given coordinates.

SetRectEmpty page 97

SetRectEmpty(*lpRect*):*nResult*

Sets the rectangle to an empty rectangle (all coordinates are zero).

SetRelAbs page 142

SetRelAbs(*hDC,nRelAbsMode*):*nOldRelAbsMode*

Sets the relabs flag.

SetResourceHandler page 216

SetResourceHandler(*hInstance,lpType,lpLoadFunc*):*lpLoadFunc*

Sets the function address of the resource handler for resources with type *lpType*. A resource handler provides for loading of custom resources.

SetROP2 page 144

SetROP2(*hDC,nDrawMode*):*nOldDrawMode*

Sets the current drawing mode.

SetScrollPos page 79

SetScrollPos(*hWnd,nBar,nPos,bRedraw*):*nOldPos*

Sets scroll bar elevator to *nPos*; redraws scroll bar if *bRedraw* is nonzero.

SetScrollRange page 80

SetScrollRange(*hWnd,nBar,nMinPos,nMaxPos,bRedraw*)

Sets minimum and maximum scroll bar positions for given scroll bar.

SetSoundNoise page 238

SetSoundNoise(*nSource,nDuration*):*nResult*

Sets the source and duration of a noise from the play device.

SetStretchBltMode page 146

SetStretchBltMode(*hDC,nStretchMode*):*nOldStretchMode*

Sets the stretching mode for the **StretchBlt** function.

SetSysColors page 102

SetSysColors(*nChanges,lpSysColor,lpColorValues*)

Changes one or more system colors.

SetSysModalWindow page 86

SetSysModalWindow(*hWnd*):*hPrevWnd*

Makes the specified window a system-modal window.

SetTextCharacterExtra page 167

SetTextCharacterExtra(*hDC,nCharExtra*):*nOldCharExtra*

Sets the amount of intercharacter spacing.

SetTextColor page 144

SetTextColor(*hDC,rgbColor*):*rgbOldColor*

Sets text color to the device color closest to *rgbColor*.

SetTextJustification page 165

SetTextJustification(*hDC,nBreakExtra,nBreakCount*):*nSet*

Prepares GDI to justify a text line using *nBreakExtra* and *nBreakCount*.

SetTimer page 59

SetTimer(*hWnd,nIDEvent,wElapse,lpTimerFunc*):*nIDNewEvent*

Creates system timer event identified by *nIDEvent*. *wElapse* is elapsed milliseconds. *lpTimerFunc* receives timer messages; if NULL, messages go to application queue.

SetViewportExt page 156

SetViewportExt(*hDC,X,Y*):*ptOldExtents*

Sets the *x*- and *y*-extents of the viewport of the specified display context.

SetViewportOrg page 153

SetViewportOrg(*hDC,X,Y*):*ptOldOrigin*

Sets the viewport origin of the specified display context.

SetVoiceAccent page 236

SetVoiceAccent(*nVoice,nTempo,nVolume,nMode,nPitch*):*nResult*

Places an accent (tempo, volume, mode, and pitch) in the voice queue *nVoice*.

SetVoiceEnvelope page 237

SetVoiceEnvelope(*nVoice,nShape,nRepeat*):*nResult*

Places the envelope (wave shape and repeat count) in the voice queue *nVoice.*

SetVoiceNote page 238

SetVoiceNote(*nVoice,nValue,nLength,nCdots*):*nResult*

Places a note in the voice queue *nVoice.*

SetVoiceQueueSize page 235

SetVoiceQueueSize(*nVoice,nBytes*):*nResult*

Allocates *nBytes* of memory for the voice queue *nVoice.* Default is 192 bytes.

SetVoiceSound page 239

SetVoiceSound(*nVoice,nFrequency,nDuration*):*nResult*

Places a sound (frequency and duration) in the voice queue *nVoice.*

SetVoiceThreshold page 242

SetVoiceThreshold(*nVoice,nNotes*):*nResult*

Sets the threshold level to *nNotes* for the voice queue *nVoice.*

SetWindowExt page 152

SetWindowExt(*hDC,X,Y*):*ptOldExtents*

Sets the *x*- and *y*-extents of the window of the specified display context.

SetWindowLong page 32

SetWindowLong(*hWnd,nIndex,lNewLong*):*lOldLong*

Changes the window attribute identified by *nIndex.*

SetWindowOrg page 151

SetWindowOrg(*hDC,X,Y*):*ptOldOrigin*

Sets the window origin of the specified display context.

SetWindowsHook page 103

SetWindowsHook(*nFilterType,lpFilterFunc***):***lpPrevFilterFunc*

Installs a system and/or application hook function.

SetWindowText page 84

SetWindowText(*hWnd,lpString***)**

Sets window caption (if any) or text (if a control) to *lpString.*

SetWindowWord page 31

SetWindowWord(*hWnd,nIndex,wNewWord***):***wOldWord*

Changes the window attribute specified by *nIndex.*

ShowCaret page 94

ShowCaret(*hWnd***)**

Displays newly-created caret or redisplays hidden caret.

ShowCursor page 92

ShowCursor(*bShow***):***nCount*

Adds 1 to cursor display count if *bShow* is nonzero. Subtracts 1 if *bShow* is zero.

ShowWindow page 33

ShowWindow(*hWnd,nCmdShow***):***bShown*

Displays or removes the given window as specified by *nCmdShow.*

SizeofResource page 216

SizeofResource(*hInstance,hResInfo***):***wBytes*

Returns the size, in bytes, of resource *hResInfo.*

StartSound page 240

StartSound( ):*nResult*

Starts play in each voice queue.

StopSound page 240

StopSound( ):*nResult*

Stops playing all voice queues, and flushes the contents of the queues.

StretchBlt page 117

StretchBlt(*hDestDC,X,Y,nWidth,nHeight,hSrcDC,XSrc,YSrc,nSrcWidth,nSrcHeight,dwRop***):***bDrawn*

Moves bitmap from source rectangle into destination rectangle, stretching or compressing as necessary. Source origin is at *XSrc, YSrc*. *X, Y, nWidth,* and *nHeight* give origin and dimensions of rectangle on destination device. *dwRop* defines how source and destination bits are combined.

SwapMouseButton see Update

SwapMouseButton(*bSwap***):***bSwapped*

Swaps the meaning of the left and right mouse buttons if *bSwap* is TRUE.

SyncAllVoices page 241

SyncAllVoices( ):*nResult*

Places a sync mark in each voice queue. Voices wait at the sync mark until all queues have encountered it.

TextOut page 119

TextOut(*hDC,X,Y,lpString,nCount***):***bDrawn*

Writes character string using current font and starting at *X, Y*.

Throw page 195

Throw(*lpCatchBuf,nThrowBack***)**

Restores the execution environment to the values in buffer *lpCatchBuf*. Execution continues at the location specified by the environment with the return value *nThrowBack* available for processing.

TranslateAccelerator page 16

TranslateAccelerator(*hWnd,hAccTable,lpMsg***):***nTranslated*

Processes keyboard accelerators for menu commands.

TranslateMessage page 15

TranslateMessage(*lpMsg*):*bTranslated*

Translates virtual keystroke messages into character messages.

TransmitCommChar page 228

TransmitCommChar(*nCid,cChar*):*nResult*

Places the character *cChar* at the head of the transmit queue for immediate transmission.

UngetCommChar page 228

UngetCommChar(*nCid,cChar*):*nResult*

Makes the character *cChar* the next character to be read from the receive queue.

UnionRect page 98

UnionRect(*lpDestRect,lpSrc1Rect,lpSrc2Rect*):*nUnion*

Stores the union of two rectangles at *lpDestRect*.

UnlockData page 205

UnlockData(*Dummy*)

Unlocks the data segment.

UnlockSegment see Update

UnlockSegment(*wSegment*):*hMem*

Unlocks the segment whose segment address is *wSegment*.

UnrealizeObject page 158

UnrealizeObject(*hBrush*):*bUnrealized*

Directs GDI to reset the origin of the given brush the next time it is selected.

UpdateWindow page 74

UpdateWindow(*hWnd*)

Notifies application when parts of a window need redrawing after changes.

ValidateRect page 76

ValidateRect(*hWnd,lpRect*)

Releases from repainting rectangle specified by *lpRect* (in client coordinates). If *lpRect* is NULL, entire window is validated.

ValidateRgn page 77

ValidateRgn(*hWnd,hRgn*)

Releases *hRgn* from repainting. If *hRgn* is NULL, entire region is validated.

WaitMessage page 13

WaitMessage( )

Yields control to other applications when application has no tasks to perform.

WaitSoundState page 240

WaitSoundState(*nState*):*nResult*

Waits until the play driver enters the state *nState.*

WindowFromPoint page 96

WindowFromPoint(*Point*):*hWnd*

Identifies the window containing *Point* (in screen coordinates).

WinMain page 9

WinMain(*hInstance,hPrevInstance,lpCmdLine,nCmdShow*):*nExitCode*

Serves as entry point for execution of a Windows application.

WndProc page 20

WndProc(*hWnd,wMsg,wParam,lParam*):*lReply*

Processes messages sent to it by Windows or the application's main function.

WriteComm page 227

WriteComm(*nCid,lpBuf,nSize*):*nBytes*

Writes up to *nSize* bytes from buffer *lpBuf* to communication device *nCid.*

WriteProfileString page 224

WriteProfileString(*lpApplicationName,lpKeyName,lpString***):***bResult*

Copies character string *lpString* to the *win.ini* file. The string replaces the current string named by *lpKeyName* in section *lpSectionName*. If the key or section does not exist, a new key and section are created.

Yield page 206

Yield( ):*bResult*

Halts the current task and starts any waiting task.

Windows Message Index

Index to Programmer's Reference

* See the *Microsoft Windows Update to Programmer's Reference and Programming Guide*

Index to Programmer's Reference (continued)

* See the *Microsoft Windows Update to Programmer's Reference and Programming Guide*

Window Messages

Message Parameters

Message	wParam	lParam (low/high)
WM_ACTIVATE	activation type[a]	window handle[b]/iconic flag[c]
WM_ACTIVATEAPP	activation flag[d]	task handle[e]
WM_ASKCBFORMATNAME	bytes to copy	**LPSTR** to buffer
WM_CANCELMODE	-	-
WM_CHANGECBCHAIN	window handle[f]	window handle[g]
WM_CHAR	ASCII value	key state[h]
WM_CLOSE	-	-
WM_COMMAND	command ID[i]	command type[j]
WM_CREATE	-	**LPCREATESTRUCT**
WM_CTLCOLOR	**HDC** to child	child handle/control type[k]
WM_DEADCHAR	dead key value	key state[h]
WM_DESTROY	-	-
WM_DESTROYCLIPBOARD	-	-
WM_DEVMODECHANGE	-	**LPSTR** to device name
WM_DRAWCLIPBOARD	-	-
WM_ENABLE	disabled flag[l]	-
WM_ENDSESSION	end session flag[m]	-

a - Activation type is zero if inactivated, 1 if activated by non-mouse, 2 if activated by mouse.

b - Low-order word is handle of window being inactivated if activation type is 1 or 2, otherwise, it is handle to window being activated.

c - High-order word is nonzero if window iconic, zero otherwise.

d - Activation flag is nonzero if application is being activated, zero otherwise.

e - Low-order word is handle to task being inactivated if activation flag is nonzero, otherwise it is handle to task being activated.

f - Handle to window being removed from chain.

g - Low-order word is handle to window following the removed window.

h - Bits 1-16: repeat count; bits 17-25: OEM scan code; bit 29: 1, if with ALT key, zero if not; bit 30: 1 if key pressed before, zero if not; bit 31: 1 if key released, zero if pushed.

i - Menu item ID, control ID, or accelerator ID.

j - Zero if menu item; 1 in high-order word if accelerator key; window handle in low-order word and notification code in high-order word if control.

k - High-order word is CTLCOLOR_MSGBOX, CTLCOLOR_EDIT, CTLCOLOR_LISTBOX, CTLCOLOR_BTN, CTLCOLOR_DLG, CTLCOLOR_SCROLLBAR, or CTLCOLOR_STATIC.

l - Disabled flag is nonzero if window is disabled, zero otherwise.

m - End session flag is nonzero if session ending, zero if continuing.

Message Parameters (continued)

Message	wParam	lParam (low/high)
WM_ERASEBKGND[a]	**HDC** to window	-
WM_FONTCHANGE	-	-
WM_GETDLGCODE[b]	-	-
WM_GETTEXT[c]	max. byte count	**LPSTR** to buffer
WM_GETTEXTLENGTH[d]	-	-
WM_HSCROLL	scroll code[e]	thumb position[f]
WM_HSCROLLCLIPBOARD	window handle[g]	scroll code[h]
WM_INITDIALOG[i]	control handle[j]	-
WM_INITMENU	menu handle	-
WM_INITMENUPOPUP	menu handle	item index/system menu[k]
WM_KEYDOWN	VK_ key code	key state[l]
WM_KEYUP	VK_ key code	key state[k]
WM_KILLFOCUS	window handle[m]	-
WM_LBUTTONDBLCLK	key state[n]	**POINT**[o]
WM_LBUTTONDOWN	key state[n]	**POINT**[o]
WM_LBUTTONUP	key state[n]	**POINT**[o]
WM_MBUTTONDBLCLK	key state[n]	**POINT**[o]
WM_MBUTTONDOWN	key state[n]	**POINT**[o]

a - Must return nonzero if background erased, zero otherwise.

b - Must return DLGC_WANTARROWS, DLGC_WANTTAB, DLGC_WANTALLKEYS, or DLGC_HASSETSEL.

c - Must return number of bytes copied.

d - Must return number of bytes in title text.

e - SB_LINEUP, SB_LINEDOWN, SB_PAGEUP, SB_PAGEDOWN, SB_THUMBPOSITION, SB_THUMBTRACK, SB_TOP, SB_BOTTOM, or SB_ENDSCROLL.

f - Thumb position in low-order word for SB_THUMBPOSITION and SB_THUMBTRACK only.

g - Handle to Clipboard application window (*clipbrd.exe*).

h - SB_LINEUP, SB_LINEDOWN, SB_PAGEUP, SB_PAGEDOWN, SB_THUMBPOSITION, SB_TOP, SB_BOTTOM, or SB_ENDSCROLL in low-order word, and thumb position in high-order word if SB_THUMBPOSITION.

i - Must return nonzero to give first control input focus, otherwise, must return zero.

j - Handle to first control that can receive input focus.

k - High-order word is nonzero if the menu is the System menu, zero otherwise.

l - Bits 1-16: repeat count; bits 17-25: OEM scan code; bit 29: 1 if with ALT key, zero if not; bit 30: 1 if key pressed before, zero if not; bit 31: 1 if key released, zero if pushed.

m - Handle to window gaining the input focus.

n - A combination of MK_RBUTTON, MK_LBUTTON, MK_MBUTTON, MK_SHIFT, and MK_CONTROL.

o - Mouse position in client coordinates.

Message Parameters (continued)

Message	wParam	lParam (low/high)
WM_MBUTTONUP	key state[a]	POINT[b]
WM_MOUSEMOVE	key state[a]	POINT[b]
WM_MOVE	-	POINT[c]
WM_NCACTIVATE	caption flag[d]	-
WM_NCCALCSIZE	-	LPRECT[e]
WM_NCCREATE[f]	window handle	LPCREATESTRUCT
WM_NCDESTROY	-	-
WM_NCHITTEST[g]	-	POINT
WM_NCLBUTTONDBLCLK	hit type[h]	POINT[i]
WM_NCLBUTTONDOWN	hit type[h]	POINT[i]
WM_NCLBUTTONUP	hit type[h]	POINT[i]
WM_NCMBUTTONDBLCLK	hit type[h]	POINT[i]
WM_NCMBUTTONDOWN	hit type[h]	POINT[i]
WM_NCMBUTTONUP	hit type[h]	POINT[i]
WM_NCMOUSEMOVE	hit type[h]	POINT[i]
WM_NCPAINT	-	-
WM_NCRBUTTONDBLCLK	hit type[h]	POINT[i]
WM_NCRBUTTONDOWN	hit type[h]	POINT[i]
WM_NCRBUTTONUP	hit type[h]	POINT[i]
WM_PAINT	-	LPPAINTSTRUCT
WM_PAINTCLIPBOARD	window handle[j]	LPPAINTSTRUCT
WM_QUERYENDSESSION[k]	-	-
WM_QUERYOPEN[l]	-	-
WM_QUIT	exit code	-

a - A combination of MK_RBUTTON, MK_LBUTTON, MK_MBUTTON, MK_SHIFT, and MK_CONTROL.

b - Mouse position in client coordinates.

c - Screen coordinates of window's upper-left corner.

d - Caption flag is nonzero if the caption is active, zero if inactive.

e - Screen coordinates of window rectangle.

f - Must return nonzero if non-client area created, zero otherwise.

g - Must return HTNOWHERE, HTERROR, HTTRANSPARENT, HTCLIENT, HTCAPTION, HTSYSMENU, HTGROWBAR, HTMENU, HTHSCROLL, or HTVSCROLL.

h - HTNOWHERE, HTERROR, HTTRANSPARENT, HTCLIENT, HTCAPTION, HTSYSMENU, HTGROWBAR, HTMENU, HTHSCROLL, or HTVSCROLL.

i - Mouse position in screen coordinates.

j - Handle to Clipboard application window (*clipbrd.exe*).

k - Must return nonzero to continue end of session, zero to prevent it.

l - Must return nonzero to open icon, zero otherwise.

Message Parameters (continued)

Message	wParam	lParam (low/high)
WM_RBUTTONDBLCLK	key state[a]	**POINT**[b]
WM_RBUTTONDOWN	key state[a]	**POINT**[b]
WM_RBUTTONUP	key state[a]	**POINT**[b]
WM_RENDERALLFORMATS	-	-
WM_RENDERFORMAT	format type[c]	-
WM_SETFOCUS	window handle[d]	-
WM_SETREDRAW	redraw flag	-
WM_SETTEXT	-	**LPSTR** to text
WM_SETVISIBLE	show flag[e]	-
WM_SHOWWINDOW	show flag[e]	show type[f]
WM_SIZE	size type[g]	width/height
WM_SIZECLIPBOARD	window handle[h]	**LPRECT**
WM_SYSCHAR	VK_ key code	key state[i]
WM_SYSCOLORCHANGE	-	-
WM_SYSCOMMAND	command ID[j]	-
WM_SYSDEADCHAR	dead key code	key state[i]
WM_SYSKEYDOWN	VK_ key code	key state[i]
WM_SYSKEYUP	VK_ key code	key state[i]
WM_SYSTEMERROR	8 (out-of-memory)	-
WM_SYSTIMER	-	-

a - A combination of MK_RBUTTON, MK_LBUTTON, MK_MBUTTON, MK_SHIFT, and MK_CONTROL.

b - Mouse position in client coordinates.

c - CF_TEXT, CF_BITMAP, CF_TEXT, CF_BITMAP, CF_METAFILEPICT, CF_SYLK, CF_DIF, or a private type.

d - Handle to window losing the input focus.

e - Show flag is nonzero if the window is shown, zero if hidden.

f - Zero if **ShowWindow** function called, otherwise SW_OTHERZOOM, SW_OTHERUNZOOM, SW_PARENTCLOSING, or SW_PARENTOPENING.

g - SIZEICONIC, SIZEFULLSCREEN, SIZENORMAL, SIZEZOOMSHOW, or SIZEZOOMHIDE.

h - Handle to Clipboard application window (*clipbrd.exe*).

i - Bits 1-16: repeat count; bits 17-25: OEM scan code; bit 29: 1 if with ALT key, zero if not; bit 30: 1 if key pressed before, zero if not; bit 31: 1 if key released, zero if pushed.

j - SC_SIZE, SC_MOVE, SC_ICON, SC_ZOOM, SC_CLOSE, SC_NEXTWINDOW, SC_PREVWINDOW, SC_VSCROLL, SC_HSCROLL, SC_MOUSEMENU, or SC_KEYMENU.

Message Parameters (continued)

Message	wParam	lParam (low/high)
WM_ TIMECHANGE	-	-
WM_ TIMER	timer ID	**FARPROC**[a]
WM_ VSCROLL	scroll code[b]	thumb position[c]
WM_ VSCROLLCLIPBOARD	window handle[d]	scroll code[e]
WM_ WININICHANGE	-	**LPSTR** to section name[f]

a - Long pointer to timer call-back function.

b - SB_ LINEUP, SB_ LINEDOWN, SB_ PAGEUP, SB_ PAGEDOWN, SB_ THUMBPOSITION, SB_ THUMBTRACK, SB_ TOP, SB_ BOTTOM, or SB_ ENDSCROLL.

c - Thumb position in low-order word for SB_ THUMBPOSITION and SB_ THUMBTRACK only.

d - Handle to Clipboard application window (*clipbrd.exe*).

e - SB_ LINEUP, SB_ LINEDOWN, SB_ PAGEUP, SB_ PAGEDOWN, SB_ THUMBPOSITION, SB_ TOP, SB_ BOTTOM, or SB_ ENDSCROLL in low-order word, and thumb position in high-order word if SB_ THUMBPOSITION.

f - NULL if more than one section changed.

Button-Control Messages

Message Parameters

Message	wParam	lParam (low/high)
BM_GETCHECK[a]	-	-
BM_GETSTATE[b]	-	-
BM_SETCHECK	check flag[c]	-
BM_SETSTATE	state flag[d]	-

a - Returns zero if not checked, 1 if checked, 2 if grayed (3-state only).
b - Returns zero if no highlight, 1 if highlight.
c - If zero, clear check. If 1, set check. If 2, gray check (3-state only).
d - If 1, set highlight. If zero, clear highlight.

Edit-Control Messages

Message Parameters

Message	wParam	lParam (low/high)
EM_CANUNDO[a]	-	-
EM_FMTLINES[b]	format flag	-
EM_GETHANDLE[c]	-	-
EM_GETLINE[d]	line number	**LPSTR**[e]
EM_GETLINECOUNT[f]	-	-
EM_GETMODIFY[g]	-	-
EM_GETRECT	-	**LPRECT**[h]
EM_GETSEL[i]	-	-
EM_LIMITTEXT	max. bytes	-
EM_LINEINDEX[j]	line number	-
EM_LINELENGTH[k]	line number	-
EM_LINESCROLL	-	line scroll/character scroll
EM_REPLACESEL	-	**LPSTR**[l]
EM_SETFONT	font ID[m]	-
EM_SETHANDLE	buffer handle[n]	-
EM_SETMODIFY	modify flag[o]	-
EM_SETRECT	-	**LPRECT**[p]
EM_SETRECTNP	-	**LPRECT**[q]

a - Returns TRUE if can undo last change.
b - Returns TRUE if text formatted.
c - Returns handle to text buffer (relative to local heap).
d - Returns number of lines in text.
e - Long pointer to buffer to receive text.
f - Returns number of lines of text.
g - Returns state of modify flag.
h - Long pointer to buffer to receive rectangle.
i - Returns start- and end-character positions of the selection.
j - Returns character position of line.
k - Returns character length of line.
l - Long pointer to replacement string.
m - Font constant from **GetStockObject** function.
n - Handle to text buffer (relative to local heap).
o - Must be TRUE to set modify flag.
p - Long pointer to rectangle.
q - Long pointer to rectangle.

Message Parameters (continued)

Message	wParam	lParam (low/high)
EM_SETSEL	-	start pos./end pos.
EM_UNDO[a]	-	-
WM_CLEAR	-	-
WM_COPY	-	-
WM_CUT	-	-
WM_UNDO	-	-

a - Returns TRUE if last change restored.

List-Box Messages

Message Parameters

Message	wParam	lParam (low/high)
LB_ADDSTRING[a]	-	**LPSTR**[b]
LB_DELETESTRING[c]	index	-
LB_DIR[d]	DOS attributes	**LPSTR**[e]
LB_GETCOUNT[f]	-	-
LB_GETCURSEL[g]	-	-
LB_GETSEL[h]	index	-
LB_GETTEXT[i]	index	**LPSTR**[j]
LB_GETTEXTLEN[k]	index	-
LB_INSERTSTRING[l]	index	**LPSTR**[m]
EM_RESETCONTENT	-	-
LB_SELECTSTRING[n]	index	**LPSTR**[o]
LB_SETCURSEL	index	-
LB_SETSEL	set/clear flag[p]	index/-

a - Returns index for string.
b - Long pointer to new string.
c - Returns number of strings in list box.
d - Returns number of strings added to list box.
e - Long pointer to pathname specification.
f - Returns number of strings in list box.
g - Returns index of selection.
h - Returns TRUE if string is selected.
i - Returns character length of string.
j - Long pointer to buffer to receive text.
k - Returns character length of string.
l - Returns index of inserted string.
m - Long pointer to string.
n - Returns index of selected string.
o - Long pointer to buffer to receive string.
p - If TRUE, select string.

Windows Message Numeric List

Code	Message	Code	Message
0x0001	WM_CREATE	0x00A5	WM_NCRBUTTONUP
0x0002	WM_DESTROY	0x00A6	WM_NCRBUTTONDBLCLK
0x0003	WM_MOVE	0x00A7	WM_NCMBUTTONDOWN
0x0005	WM_SIZE	0x00A8	WM_NCMBUTTONUP
0x0006	WM_ACTIVATE	0x00A9	WM_NCMBUTTONDBLCLK
0x0007	WM_SETFOCUS	0x0100	WM_KEYDOWN
0x0008	WM_KILLFOCUS	0x0101	WM_KEYUP
0x0009	WM_SETVISIBLE	0x0102	WM_CHAR
0x000A	WM_ENABLE	0x0103	WM_DEADCHAR
0x000B	WM_SETREDRAW	0x0104	WM_SYSKEYDOWN
0x000C	WM_SETTEXT	0x0105	WM_SYSKEYUP
0x000D	WM_GETTEXT	0x0106	WM_SYSCHAR
0x000E	WM_GETTEXTLENGTH	0x0107	WM_SYSDEADCHAR
0x000F	WM_PAINT	0x0110	WM_INITDIALOG
0x0010	WM_CLOSE	0x0111	WM_COMMAND
0x0011	WM_QUERYENDSESSION	0x0112	WM_SYSCOMMAND
0x0012	WM_QUIT	0x0113	WM_TIMER
0x0013	WM_QUERYOPEN	0x0114	WM_HSCROLL
0x0014	WM_ERASEBKGND	0x0115	WM_VSCROLL
0x0015	WM_SYSCOLORCHANGE	0x0116	WM_INITMENU
0x0016	WM_ENDSESSION	0x0117	WM_INITMENUPOPUP
0x0017	WM_SYSTEMERROR	0x0118	WM_SYSTIMER
0x0018	WM_SHOWWINDOW	0x0200	WM_MOUSEMOVE
0x0019	WM_CTLCOLOR	0x0201	WM_LBUTTONDOWN
0x001A	WM_WININICHANGE	0x0202	WM_LBUTTONUP
0x001B	WM_DEVMODECHANGE	0x0203	WM_LBUTTONDBLCLK
0x001C	WM_ACTIVATEAPP	0x0204	WM_RBUTTONDOWN
0x001D	WM_FONTCHANGE	0x0205	WM_RBUTTONUP
0x001E	WM_TIMECHANGE	0x0206	WM_RBUTTONDBLCLK
0x001f	WM_CANCELMODE	0x0207	WM_MBUTTONDOWN
0x0081	WM_NCCREATE	0x0208	WM_MBUTTONUP
0x0082	WM_NCDESTROY	0x0209	WM_MBUTTONDBLCLK
0x0083	WM_NCCALCSIZE	0x0305	WM_RENDERFORMAT
0x0084	WM_NCHITTEST	0x0306	WM_RENDERALLFORMATS
0x0085	WM_NCPAINT	0x0307	WM_DESTROYCLIPBOARD
0x0086	WM_NCACTIVATE	0x0308	WM_DRAWCLIPBOARD
0x0087	WM_GETDLGCODE	0x0309	WM_PAINTCLIPBOARD
0x00A0	WM_NCMOUSEMOVE	0x030A	WM_VSCROLLCLIPBOARD
0x00A1	WM_NCLBUTTONDOWN	0x030B	WM_SIZECLIPBOARD
0x00A2	WM_NCLBUTTONUP	0x030C	WM_ASKCBFORMATNAME
0x00A3	WM_NCLBUTTONDBLCLK	0x030D	WM_CHANGECBCHAIN
0x00A4	WM_NCRBUTTONDOWN	0x030E	WM_HSCROLLCLIPBOARD