WordPerfect for Windows Macro Command Quick Reference

This help file summarizes the WordPerfect for Windows macro programming and product function commands, up to and including commands new to the "May 1992" interim release (the interim release files are dated 4/30/1992).

From **WordPerfect for Windows Power Tools**, by Gordon McComb
Published by Bantam Books.

Click on one of the following to jump to that topic.

Help Using WPWin Macro Help

This WordPerfect for Windows (WPWin) quick reference is a companion help file for the book **WordPerfect for Windows Power Tools**, by Gordon McComb, published 1992 by Bantam Books. It contains on-line help on all WPWin macro commands, including programming and product function commands.

Jump on a topic by clicking on it (linked topics are shown underlined and in color). For example, to return to the index, click in the Index entry (Contents if using Windows 3.1).

You may also:

- Click the Index button to move back to the index/contents.
- Click the Back button to return to the previous help topic.

What You'll Find

This on-line help is divided into five main sections:

- **Product Functions -- Alphabetical**. Alphabetical listing of product function commands (such as **Type** and **HardReturn**).
- **Product Functions -- By Type**. Listing of product function commands, separated into major categories.
- **Programming Commands -- Alphabetical**. Alphabetical listing of programming commands (such as **Assign** and **Menu**).
- **Programming Commands -- By Type**. Listing of programming commands, separated into major categories.
- **Topics**. General topics on using WPWin macro commands.

Definition of Icons

Icons are used to identify command types, and to highlight notes within the help text:

Macro product command

Macro programming command


Product command used only when writing macros


Dialog box product command

Useful note

New command introduced in the May 1992 interim release

Product function command enhancement introduced in the May 1992 interim release.

Context Help from a Macro

This on-line help can be called from a WPWin macro using "context-sensitive" command look-up Click on Context Macro to view the macro.

To find help for a command:

1. Type the command (no spelling errors, please!). The insertion point should be next to or over the command to look up. Be sure there is at least one space between the command and any parentheses, such as AboutDlg ( ). You can also select the text of the command using the mouse or keyboard.
2. Play the macro.
3. Help for the command is displayed in the window.
4. Close the Help window when done. (You can also keep the Help window open; minimize the window to move it out of the way.)

Before using the macro you may need to edit the Filename variable to include a disk and directory path. You can avoid this step by making sure this help file is in your Windows directory.

Starting Help from a Macro

You may also access this on-line help using a Calling Macro. This macro merely starts the on-line help, and displays the main index

// Context-sensitive help macro

```
// Choose Edit-Copy to copy this macro to the clipboard.
// You can then paste into a document.

// From WordPerfect for Windows Power Tools,
// published by Bantam Computer Books

Filename:="machelp.hlp" // Edit this as necessary

Application (wp;wpwp;default;"wpwpUS.wcd")
GetWPData (IfSelectOn;SelectModeActive!)
If (NOT IfSelectOn)
      SelectWord()
EndIf
GetWPData (HelpWord;SelectedText!)
SelectMode (Off!)
StrLen (HWLength; HelpWord)
HWTemp:=""
For (Count;1;Count<=HWLength;Count+1)
      SubStr (TestString;Count;1;HelpWord)
      If (TestString <> " ")
            HWTemp:= HWTemp + TestString
      EndIf
EndFor
HelpWord:=HWTemp
DLLLoad (User; "USER")
DLLCall (User; "WinHelp"; bHelp:BOOL;{
      LOWORD(0);
      Filename;
      LOWORD(257);
      Address (HelpWord)}
)
DLLFree (User)
```

// call on-line help macro

```
// Choose Edit-Copy to copy this macro to the clipboard.
// You can then paste into a document.

// From WordPerfect for Windows Power Tools,
// published by Bantam Computer Books

Filename:="machelp.hlp" // Edit this as necessary
DLLLoad (User; "USER")
DLLCall (User; "WinHelp"; bHelp:BOOL;{
      LOWORD(0);
      Filename;
      LOWORD(3);
      ""}
)
DLLFree (User)
```

Topics

Product Function Commands -- By Type

Add-In

FileManager
Sort
SortDlg
Speller
Thesaurus
WordCountDlg

Advance

Advance
AdvanceDlg

Application and Document Window

AboutDlg
AppMaximize
AppMinimize
CloseNoSave
DocMaximize
DocMinimize
DocMove
DocNext
DocPrevious
DocRestore
DocSize
WindowCascade
WindowTile

Button Bar

ButtonBarEditDlg
ButtonBarNewDlg
ButtonBarOptionsDlg
ButtonBarOptions
ButtonBarSave
ButtonBarSaveDlg
ButtonBarSelect
ButtonBarSelectDlg
ButtonBarShow

Case Conversion

CaseToLower
CaseToUpper

Clipboard

ConvertClipboardDlg
ConvertClipboardPicture
EditAppend
EditCopy
EditCut
EditPaste

Code Insert

HardHyphen
HardPageBreak
HardPageBreakInsert
HardReturn
HardReturnInsert
HardSpace
HardTabCenter
HardTabCenterDot
HardTabDecimal
HardTabDecimalDot
HardTabDecimalInsert
HardTabLeft
HardTabLeftDot
HardTabRight
HardTabRightDot
Hyphen
HyphenIgnoreWord
HyphenSoft
HyphenSoftReturn
InsertSpecialCodesDlg
InsertTypeover

Columns

ColumnDefine
ColumnDefineDlg
ColumnDefineEven
ColumnsOff
ColumnsOn

Comments

CommentConvertToText
CommentCreate
CommentCreateDlg
CommentEditDlg
CommentShow

Cross-Reference

CrossRefMark
CrossRefMarkDlg

Cursor/Insertion Point Movement

MarginRelease
MarginReleaseInsert
PosCellDown
PosCellNext
PosCellPrevious
PosCellUp
PosCharNext
PosCharPrevious
PosColumnNext
PosColumnPrevious
PosDocBottom
PosDocTop
PosDocVeryTop
PosGoTo
PosGoToDlg
PosLineBegin
PosLineDown
PosLineEnd
PosLineUp
PosLineVeryBegin
PosLineVeryEnd
PosPageBottom
PosPageNext
PosPagePrevious
PosPageTop
PosParagraphNext
PosParagraphPrevious
PosScreenDown
PosScreenLeft
PosScreenRight
PosScreenUp
PosTableBegin
PosTableColumnBottom
PosTableColumnTop
PosTableEnd
PosTableRowBegin
PosTableRowEnd
PosWordNext
PosWordPrevious
ScrollScreenLeft
ScrollScreenRight

Date

DateCode
DateFormat
DateFormatDlg
DateText

DDE Link

DDECreateLink
DDECreateLinkDlg
DDEDeleteLink
DDEDeleteLinkDlg
DDEEditLink
DDEEditLinkDlg
DDEPasteLink
DDEUpdateLink
DDEUpdateLinkDlg

Document Compare

DocCompare
DocCompareAddMarkingsDlg
DocCompareRemove
DocCompareRemoveMarkingsDlg

Document Summary

DocSummaryDefine
DocSummaryDelete
DocSummaryDlg
DocSummaryExtract
DocSummaryGetData
DocSummaryPrint
DocSummarySaveAs

Edit

Close
CloseNoSave
DeleteCharNext
DeleteCharPrevious
DeleteEOL
DeleteEOP
DeleteToBeginningOfWord
DeleteToEndOfWord
DeleteWord
Undelete
UndeleteDlg
Undo

Endnotes/Footnotes

Close
EndnoteCreate
EndnoteEdit
EndnoteEditDlg
EndnoteNewNumberDlg
EndnoteNewNumber
EndnoteOptions
EndnoteOptionsDlg
EndnotePlacementDlg
EndnotePlacement
FootnoteCreate
FootnoteEdit
FootnoteEditDlg
FootnoteNewNumberDlg
FootnoteNewNumber
FootnoteOptions
FootnoteOptionsDlg
NoteNext
NoteNumber
NotePrevious

File Functions

Close
CloseNoSave
FileChangeDir
FileCopy
FileDelete
FileMove
FileManager
FileNew
FileOpen
FileOpenDlg
FileRetrieve
FileRetrieveDlg
FileSave
FileSaveAsDlg

Fonts

Font
FontBold
FontColor
FontColorDlg
FontDlg
FontDoubleUnderline
FontExtraLarge
FontFine
FontItalic
FontLarge
FontNormal
FontOutline
FontRedline
FontShadow
FontSmall
FontSmallCaps
FontStrikeout
FontSubscript
FontSuperscript
FontUnderline
FontVeryLarge

Formatting -- Line

ForcePageEven
ForcePageOdd
JustifyCenter
JustifyFull
JustifyLeft
JustifyRight
KerningSpacing
LineCenter
LineCenterEnd
LineDraw
LineDrawDlg
LineFlushRight
LineHeight
LineHeightDlg
LineHyphenation
LineHyphenationDlg
LineNumbering
LineNumberingDlg
LineSpacing
LineSpacingDlg
Tab
TabInsert
TabSet
TabSetDlg

Formatting -- Page

FooterDlg
HeaderDlg
HeaderFooter
HeaderFooterPlacementDlg
HeaderFooterPlacement
PageCenter
PageMargins
PageMarginsDlg
PageNumbering
PageNumberingDlg
PageNumberInsert
PageSuppress
PageSuppressDlg
PageWidowOrphan
PaperSizeAdd
PaperSizeDelete
PaperSizeDlg
PaperSizeSelect

Formatting -- Paragraph

ParagraphDoubleIndent
ParagraphHangingIndent
ParagraphIndent
ParagraphNumber
ParagraphNumberDefDlg
ParagraphNumberingDlg

Graphics

BoxEditCaption
BoxEditor
BoxFigureEdit
BoxNewNumber
BoxOptions
BoxPosition
BoxRetrieve
BoxSelect
CaptionBoxNumber
Close
EquationCaptionEdit
EquationCreate
EquationCreateDlg
EquationEdit
EquationNewNumberDlg
EquationOptionsDlg
EquationPalette
EquationPositionDlg
EquationSettingsDlg
EquationSettings
EquationZoom100
EquationZoom200
EquationZoomFill
EquationZoomIn
EquationZoomOut
FigureCaptionEdit
FigureCreate
FigureCreateDlg
FigureEdit
FigureNewNumberDlg
FigureOptionsDlg
FigurePositionDlg
FigureRetrieveDlg
GraphicLine
GraphicLineSelect
GraphicsShow
HorizLineCreateDlg
HorizLineEditDlg
TableBoxCaptionEdit
TableBoxCreate
TableBoxCreateDlg
TableBoxEdit
TableBoxNewNumberDlg
TableBoxOptionsDlg
TableBoxPositionDlg
TextBoxCaptionEdit
TextBoxCreate
TextBoxCreateDlg
TextBoxEdit
TextBoxNewNumberDlg
TextBoxOptionsDlg
TextBoxPositionDlg
TextBoxRotate
TextBoxRotateDlg

UserBoxCaptionEdit
UserBoxCreate
UserBoxCreateDlg
UserBoxEdit
UserBoxNewNumberDlg
UserBoxOptionsDlg
UserBoxPositionDlg
VertLineCreateDlg
VertLineEditDlg

Help

HelpContextSensitive
HelpGlossary
HelpHowDoI
HelpIndex
HelpKeyboard
HelpUsingWPHelp
HelpUsingMacros
HelpWhatIs

Lists, Tables, etc.

Generate
GenerateDlg
IndexDefine
IndexDefineDlg
IndexMark
IndexMarkDlg
ListDefine
ListDefineDlg
ListMark
ListMarkDlg
ToADefine
ToADefineDlg
ToAEditFull
ToAMarkFull
ToAMarkShort
ToAMarkShortDlg
ToCDefine
ToCDefineDlg
ToCMark
ToCMarkDlg

Macros, Merges, Keyboards

Display
GetWPData
KeyboardSelect
KeyboardSelectDlg
MacroAssignDlg
MacroMenuAppend
MacroMenuDelete
MacroStatusPrompt
MergeCodesDlg
MergeConvert
MergeConvertMsgDlg
MergeDOSText
MergeEndField
MergeEndRecord
MergeExecute
MergeFieldDlg
MergeFilesDlg
MergeInputDlg
MergeInsertCode
MergeNextRecord
MergePageOff
MergeVariableGet
MergeVariableSet
PauseKey
Type
TypeChar
UserFunction

Master Document

MasterDocCondense
MasterDocCondenseDlg
MasterDocExpand
MasterDocSubdocDlg
MasterDocSubdocInsert

Miscellaneous

BlockProtect
ConditionalEOP
ConditionalEOPDlg
DisplayFontAdjustment
DisplayPitch
DisplayPitchDlg
Redisplay
Zoom
ZoomDlg
Zoom100
Zoom150
Zoom200
Zoom50
Zoom75
ZoomToPageWidth

Outlining

NumberFormat
OutlineCopy
OutlineDefine
OutlineDelete
OutlineMove
OutlineOff
OutlineOn

Preferences and Setup

DocInitialCodes
DocInitialFont
DocInitialFontDlg
DocRedlineMethodDlg
DocRedlineMethod
DraftMode
HorizontalScrollBarShow
Language
LanguageDlg
PrefBackup
PrefBackupDlg
PrefBeep
PrefDateFormat
PrefDateFormatDlg
PrefDisplayDlg
PrefDisplaySet
PrefDocSummary
PrefDocSummaryDlg
PrefDraftColors
PrefDraftColorsDlg
PrefEnvSettings
PrefEnvSettingsDlg
PrefEquation
PrefEquationDlg
PrefHyphenation
PrefInitialCodes
PrefLocationDlg
PrefLocationOfFiles
PrefMenu
PrefMergeDelimit
PrefMergeDlg
PrefPrintCopies
PrefPrintDoc
PrefPrintRedline
PrefPrintSettingsDlg
PrefPrintSizeRatio
PrefPrintWindowsDriverSet
PrefRevealCodeColorsDlg
PrefRevealCodeColors
PrefRuler
PrefSave
PrefToA
PrefToADlg
PrefZoom
PrefZoomDlg
RevealCodes
RulerShow
ShortMenus

Printing

PrintDlg
PrintDoc
PrinterCommand
PrinterCommandDlg
PrinterInitialize
PrinterSelectDlg
PrintFull
PrintMultiplePages
PrintOptions
PrintPage
PrintPreview
PrintSelected

Search/Replace

SearchDlg
SearchNext
SearchPrevious
SearchReplace
SearchReplaceDlg
SearchText

Selection

SelectAll
SelectCell
SelectCellDown
SelectCellDownArrow
SelectCellLeft
SelectCellRight
SelectCellUp
SelectCellUpArrow
SelectCharNext
SelectCharPrevious
SelectColumn
SelectColumnBottom
SelectColumnNext
SelectColumnPrevious
SelectColumnTop
SelectDocBottom
SelectDocTop
SelectDocVeryTop
SelectLineBegin
SelectLineDown
SelectLineEnd
SelectLineUp
SelectLineVeryBegin
SelectLineVeryEnd
SelectMode
SelectPage
SelectPageNext
SelectPagePrevious
SelectParagraph
SelectParagraphNext
SelectParagraphPrevious
SelectRectangle
SelectScreenDown
SelectScreenLeft
SelectScreenRight
SelectScreenUp
SelectSentence
SelectSentenceNext
SelectSentencePrevious
SelectTable
SelectTableColumn
SelectTableColumnExtendLeft
SelectTableColumnExtendRight
SelectTableRow
SelectWord
SelectWordNext
SelectWordPrevious

Special Characters

OverstrikeCreate
OverstrikeCreateDlg
OverstrikeEdit
OverstrikeEditDlg
WPCharactersDlg

Spreadsheet

SpreadsheetImportDlg
SpreadsheetImportLink
SpreadsheetLinkCreateDlg
SpreadsheetLinkEditDlg
SpreadsheetLinkOptions
SpreadsheetLinkOptionsDlg
SpreadsheetLinkUpdateAll
SpreadsheetLinkUpdateAllDlg

Styles

Styles
StylesDelete
StylesDlg
StylesEdit
StylesProperties
StylesPropertiesDlg
StylesRetrieve
StylesSave

Tables

TableAppendRow
TableCalculate
TableCell
TableCellDlg
TableColumn
TableColumnDlg
TableConvert
TableConvertDlg
TableCreate
TableCreateNewDlg
TableDeleteColumn
TableDeleteDlg
TableDeleteRow
TableEditColumn
TableFormula
TableFormulaDlg
TableInsertColumn
TableInsertDlg
TableInsertRow
TableJoin
TableLineDlg
TableLines
TableOptions
TableOptionsDlg
TableRow
TableRowDlg
TableSplit
TableSplitDlg

Typeset

TypesetBaseline
TypesetDlg
TypesetJustifyLimits
TypesetKerning
TypesetLeadingAdjust
TypesetLetterspace
TypesetManualKerningDlg
TypesetUnderlineOptions
TypesetWordSpace

Product Functions -- Alphabetical

AboutDlg
Advance
AdvanceDlg
AppMaximize
AppMinimize
BlockProtect
BoxEditCaption
BoxEditor
BoxFigureEdit
BoxNewNumber
BoxOptions
BoxPosition
BoxRetrieve
BoxSelect
ButtonBarEditDlg
ButtonBarNewDlg
ButtonBarOptionsDlg
ButtonBarOptions
ButtonBarSaveDlg
ButtonBarSave
ButtonBarSelectDlg
ButtonBarSelect
ButtonBarShow
CaptionBoxNumber
CaseToLower
CaseToUpper
Close
CloseNoSave
ColumnDefineDlg
ColumnDefineEven
ColumnDefine
ColumnsOff
ColumnsOn
CommentConvertToText
CommentCreateDlg
CommentCreate
CommentEditDlg
CommentShow
ConditionalEOPDlg
ConditionalEOP
ConvertClipboardDlg
ConvertClipboardPicture
CrossRefMarkDlg
CrossRefMark
DateCode
DateFormatDlg
DateFormat
DateText
DDECreateLinkDlg
DDECreateLink
DDEDeleteLinkDlg
DDEDeleteLink
DDEEditLinkDlg
DDEEditLink

DDEPasteLink
DDEUpdateLinkDlg
DDEUpdateLink
DeleteCharNext
DeleteCharPrevious
DeleteEOL
DeleteEOP
DeleteToBeginningOfWord
DeleteToEndOfWord
DeleteWord
Display
DisplayFontAdjustment
DisplayPitchDlg
DisplayPitch
DocCompareAddMarkingsDlg
DocCompare
DocCompareRemove
DocCompareRemoveMarkingsDlg
DocInitialCodes
DocInitialFontDlg
DocInitialFont
DocMaximize
DocMinimize
DocMove
DocNext
DocPrevious
DocRedlineMethodDlg
DocRedlineMethod
DocRestore
DocSize
DocSummaryDefine
DocSummaryDelete
DocSummaryDlg
DocSummaryExtract
DocSummaryGetData
DocSummaryPrint
DocSummarySaveAs
DraftMode
EditAppend
EditCopy
EditCut
EditPaste
EndnoteCreate
EndnoteEditDlg
EndnoteEdit
EndnoteNewNumberDlg
EndnoteNewNumber
EndnoteOptionsDlg
EndnoteOptions
EndnotePlacementDlg
EndnotePlacement
EquationCaptionEdit
EquationCreateDlg
EquationCreate
EquationEdit
EquationNewNumberDlg

EquationOptionsDlg
EquationPalette
EquationPositionDlg
EquationSettingsDlg
EquationSettings
EquationZoom100
EquationZoom200
EquationZoomFill
EquationZoomIn
EquationZoomOut
FigureCaptionEdit
FigureCreateDlg
FigureCreate
FigureEdit
FigureNewNumberDlg
FigureOptionsDlg
FigurePositionDlg
FigureRetrieveDlg
FileChangeDir
FileCopy
FileDelete
FileManager
FileMove
FileNew
FileOpenDlg
FileOpen
FileRetrieveDlg
FileRetrieve
FileSaveAsDlg
FileSave
FontBold
FontColorDlg
FontColor
FontDlg
FontDoubleUnderline
FontExtraLarge
FontFine
Font
FontItalic
FontLarge
FontNormal
FontOutline
FontRedline
FontShadow
FontSmallCaps
FontSmall
FontStrikeout
FontSubscript
FontSuperscript
FontUnderline
FontVeryLarge
FooterDlg
FootnoteCreate
FootnoteEditDlg
FootnoteEdit
FootnoteNewNumberDlg

FootnoteNewNumber
FootnoteOptionsDlg
FootnoteOptions
ForcePageEven
ForcePageOdd
GenerateDlg
Generate
GetWPData
GraphicLine
GraphicLineSelect
GraphicsShow
HardHyphen
HardPageBreak
HardPageBreakInsert
HardReturn
HardReturnInsert
HardSpace
HardTabCenterDot
HardTabCenter
HardTabDecimalDot
HardTabDecimal
HardTabDecimalInsert
HardTabLeftDot
HardTabLeft
HardTabRightDot
HardTabRight
HeaderDlg
HeaderFooter
HeaderFooterPlacementDlg
HeaderFooterPlacement
HelpContextSensitive
HelpGlossary
HelpHowDoI
HelpIndex
HelpKeyboard
HelpUsingMacros
HelpUsingWPHelp
HelpWhatIs
HorizLineCreateDlg
HorizLineEditDlg
HorizontalScrollBarShow
Hyphen
HyphenIgnoreWord
HyphenSoft
HyphenSoftReturn
IndexDefineDlg
IndexDefine
IndexMarkDlg
IndexMark
InsertSpecialCodesDlg
InsertTypeover
JustifyCenter
JustifyFull
JustifyLeft
JustifyRight
KerningSpacing

KeyboardSelectDlg
KeyboardSelect
LanguageDlg
Language
LineCenterEnd
LineCenter
LineDrawDlg
LineDraw
LineFlushRight
LineHeightDlg
LineHeight
LineHyphenationDlg
LineHyphenation
LineNumberingDlg
LineNumbering
LineSpacingDlg
LineSpacing
ListDefineDlg
ListDefine
ListMarkDlg
ListMark
MacroAssignDlg
MacroMenuAppend
MacroMenuDelete
MacroStatusPrompt
MarginRelease
MarginReleaseInsert
MasterDocCondenseDlg
MasterDocCondense
MasterDocExpand
MasterDocSubdocDlg
MasterDocSubdocInsert
MergeCodesDlg
MergeConvert
MergeConvertMsgDlg
MergeDOSText
MergeEndField
MergeEndRecord
MergeExecute
MergeFieldDlg
MergeFilesDlg
MergeInputDlg
MergeInsertCode
MergeNextRecord
MergePageOff
MergeVariableGet
MergeVariableSet
NoteNext
NoteNumber
NotePrevious
NumberFormat
OutlineCopy
OutlineDefine
OutlineDelete
OutlineMove
OutlineOff

OutlineOn
OverstrikeCreateDlg
OverstrikeCreate
OverstrikeEditDlg
OverstrikeEdit
PageCenter
PageMarginsDlg
PageMargins
PageNumberingDlg
PageNumbering
PageNumberInsert
PageSuppressDlg
PageSuppress
PageWidowOrphan
PaperSizeAdd
PaperSizeDelete
PaperSizeDlg
PaperSizeSelect
ParagraphDoubleIndent
ParagraphHangingIndent
ParagraphIndent
ParagraphNumberDefDlg
ParagraphNumber
ParagraphNumberingDlg
PauseKey
PosCellDown
PosCellNext
PosCellPrevious
PosCellUp
PosCharNext
PosCharPrevious
PosColumnNext
PosColumnPrevious
PosDocBottom
PosDocTop
PosDocVeryTop
PosGoToDlg
PosGoTo
PosLineBegin
PosLineDown
PosLineEnd
PosLineUp
PosLineVeryBegin
PosLineVeryEnd
PosPageBottom
PosPageNext
PosPagePrevious
PosPageTop
PosParagraphNext
PosParagraphPrevious
PosScreenDown
PosScreenLeft
PosScreenRight
PosScreenUp
PosTableBegin
PosTableColumnBottom

PosTableColumnTop
PosTableEnd
PosTableRowBegin
PosTableRowEnd
PosWordNext
PosWordPrevious
PrefBackupDlg
PrefBackup
PrefBeep
PrefDateFormatDlg
PrefDateFormat
PrefDisplayDlg
PrefDisplaySet
PrefDocSummaryDlg
PrefDocSummary
PrefDraftColorsDlg
PrefDraftColors
PrefEnvSettingsDlg
PrefEnvSettings
PrefEquationDlg
PrefEquation
PrefHyphenation
PrefInitialCodes
PrefLocationDlg
PrefLocationOfFiles
PrefMenu
PrefMergeDelimit
PrefMergeDlg
PrefPrintCopies
PrefPrintDoc
PrefPrintRedline
PrefPrintSettingsDlg
PrefPrintSizeRatio
PrefPrintWindowsDriverSet
PrefRevealCodeColorsDlg
PrefRevealCodeColors
PrefRuler
PrefSave
PrefToADlg
PrefToA
PrefZoom
PrefZoomDlg
PrintDlg
PrintDoc
PrinterCommandDlg
PrinterCommand
PrinterInitialize
PrinterSelectDlg
PrintFull
PrintMultiplePages
PrintOptions
PrintPage
PrintPreview
PrintSelected
Redisplay
RevealCodes

RulerShow
ScrollScreenLeft
ScrollScreenRight
SearchDlg
SearchNext
SearchPrevious
SearchReplaceDlg
SearchReplace
SearchText
SelectAll
SelectCellDownArrow
SelectCellDown
SelectCell
SelectCellLeft
SelectCellRight
SelectCellUpArrow
SelectCellUp
SelectCharNext
SelectCharPrevious
SelectColumnBottom
SelectColumn
SelectColumnNext
SelectColumnPrevious
SelectColumnTop
SelectDocBottom
SelectDocTop
SelectDocVeryTop
SelectLineBegin
SelectLineDown
SelectLineEnd
SelectLineUp
SelectLineVeryBegin
SelectLineVeryEnd
SelectMode
SelectPage
SelectPageNext
SelectPagePrevious
SelectParagraph
SelectParagraphNext
SelectParagraphPrevious
SelectRectangle
SelectScreenDown
SelectScreenLeft
SelectScreenRight
SelectScreenUp
SelectSentence
SelectSentenceNext
SelectSentencePrevious
SelectTableColumnExtendLeft
SelectTableColumnExtendRight
SelectTableColumn
SelectTable
SelectTableRow
SelectWord
SelectWordNext
SelectWordPrevious

ShortMenus
Sort
SortDlg
Speller
SpreadsheetImportDlg
SpreadsheetImportLink
SpreadsheetLinkCreateDlg
SpreadsheetLinkEditDlg
SpreadsheetLinkOptionsDlg
SpreadsheetLinkOptions
SpreadsheetLinkUpdateAllDlg
SpreadsheetLinkUpdateAll
StylesDelete
StylesDlg
StylesEdit
Styles
StylesPropertiesDlg
StylesProperties
StylesRetrieve
StylesSave
Tab
TabInsert
TableAppendRow
TableBoxCaptionEdit
TableBoxCreateDlg
TableBoxCreate
TableBoxEdit
TableBoxNewNumberDlg
TableBoxOptionsDlg
TableBoxPositionDlg
TableCalculate
TableCellDlg
TableCell
TableColumnDlg
TableColumn
TableConvertDlg
TableConvert
TableCreate
TableCreateNewDlg
TableDeleteColumn
TableDeleteDlg
TableDeleteRow
TableEditColumn
TableFormulaDlg
TableFormula
TableInsertColumn
TableInsertDlg
TableInsertRow
TableJoin
TableLineDlg
TableLines
TableOptionsDlg
TableOptions
TableRowDlg
TableRow
TableSplitDlg

TableSplit
TabSetDlg
TabSet
TextBoxCaptionEdit
TextBoxCreateDlg
TextBoxCreate
TextBoxEdit
TextBoxNewNumberDlg
TextBoxOptionsDlg
TextBoxPositionDlg
TextBoxRotateDlg
TextBoxRotate
Thesaurus
ToADefineDlg
ToADefine
ToAEditFull
ToAMarkFull
ToAMarkShortDlg
ToAMarkShort
ToCDefineDlg
ToCDefine
ToCMarkDlg
ToCMark
TypeChar
Type
TypesetBaseline
TypesetDlg
TypesetJustifyLimits
TypesetKerning
TypesetLeadingAdjust
TypesetLetterspace
TypesetManualKerningDlg
TypesetUnderlineOptions
TypesetWordSpace
UndeleteDlg
Undelete
Undo
UserBoxCaptionEdit
UserBoxCreateDlg
UserBoxCreate
UserBoxEdit
UserBoxNewNumberDlg
UserBoxOptionsDlg
UserBoxPositionDlg
UserFunction
VertLineCreateDlg
VertLineEditDlg
WindowCascade
WindowTile
WordCountDlg
WPCharactersDlg
Zoom
Zoom100
Zoom150
Zoom200
Zoom50

Zoom75
ZoomDlg
ZoomToPageWidth

Programming Commands -Alphabetical

EndFor
EndIf
EndPrompt
EndWhile
ErrorOff
ErrorOn
False
For
ForEach
Fraction
GetNumber
GetString
GetUnits
Go
HiWord
If
Inches
Integer
Label
Length
Letter
LoWord
Menu
Mod
NewDefault
Not
NumStr
OemString
OnCancel
OnCancel Call
OnError
OnError Call
OnNotFound
OnNotFound Call
Or
Pause
Points
Prompt
Quit
Real
Repeat
Return
ReturnCancel
ReturnError
ReturnNotFound
Run
Speed
String
StrLen
StrNum
StrPos
SubByte
SubStr
True
Until
Wait

While
Word
WpString
WpUnits
Xor

Anatomy of a Product Function Command

The definitions for product function commands are stored in the WPWPxx.WCD file, included in the WPC shared directory. The xx is a unique two-letter language identifier; all versions of WPWin come with the WPWPUS.WCD file, which is the US-English language version.

All product function commands follow the same basic syntax:

- Command name. Always one word; mixed capitalization is used to make the command more readable, such as in **FootnoteOptionsDlg** or **PageSuppress**.
- Argument list. Enclosed in parentheses. Not all product function commands require arguments, but they all use the parentheses.

The formal argument list of a product command consists of two parts: a parameter name, and a parameter member set value. The parameter name identifies the purpose of the parameter, as in State or FootnoteNumber. The member set value is the value of the parameter. The value may turn something on or off, or it may specify a string to use.

For example, in the following,

```
FootnoteNewNumber(
      FootnoteNumber:string
)
```

FootnoteNewNumber is the name of the command. FootnoteNumber is the parameter name, and string is the member set value, in this case a string of characters.

WPWin accepts two general forms of member set values: predefined (either names or numeric values), or user-defined (strings or numbers). Predefined names are uniquely identified by WPWin by the use of the ! character. For example, **Yes!** and **No!** are member set values that tell WPWin how to apply a certain command.

WPWin allows for numeric value aliases (called enumerated values) for all pre-defined names. For instance, you can always use 0 for **No!** and 1 for **Yes!**. This makes it easier to write macros that may be used with different language versions of WPWin. You can mix and match numbers and names in the same command, though for readability you'll probably want to stick with one approach.

There are five common types of member set values:

- **On/Off** states -- Specifies either Off! (0) or On! (1).
- **WPUnits** -- A value expressed in any valid WordPerfect units of measure, such as 1", 100.25w, 72p, 15c, and so forth.
- **Variable** -- Indicates the function accepts or returns a variable.
- **String** -- Specifies a text string, which is always enclosed in quotes. There are several string sub-types: WordPerfect, OEM (DOS), and ANSI (Windows). For simplicity this reference doesn't differentiate between them. You can always refer to the WPWPxx.WCD file for the specific string type used by the variable.
- **Numeric value** -- A number of some sort, either an integer (whole number) or real value (such as 123.45). For example, you'd use an integer to indicate a new starting page number, and a real value for the size of a graphic box.

Understanding the Formatting Syntax

The formatting of the product function commands in this reference closely parallels that of the WPWPxx.WCD file, from which the definitions were drawn. The formatting uses the { and } characters to enclose a group or option for a single parameter name.

Do not type the { or } characters; they are used only to show you that you are to use one (and only one) member set value for any given parameter name. For example, the **SpreadsheetLinkOptions** command uses two parameters, as shown here:

```
SpreadsheetLinkOptions(
        UpdateOnRetrieve:{
                No!=0;
                Yes!=1
        };
        ShowLinkCodes:{
                No!=0;
                Yes!=1
        }
)
```

This command allows you to set linking options to spreadsheet files. In a working macro, you might choose to not update the link when it is retrieved, but to show the link codes in the document. The command in the macro would appears as:

SpreadsheetLinkOptions (UpdateOnRetrieve:No!;ShowLinkCodes:Yes!)

Or, using the enumerated values instead of the names:

SpreadsheetLinkOptions (UpdateOnRetrieve:0;ShowLinkCodes:1)

Optional Use of Parameter Names

WPWin doesn't force you to include the parameter names in the formal parameter list. In fact, you will probably find it easier to omit the names. Example: The **SpreadsheetLinkOptions** command can be written in either of the following two ways:

SpreadsheetLinkOptions (No!;Yes!)

or

SpreadsheetLinkOptions (0;1)

WPWin does impose a few rules about omitting parameter names:

- If you use a parameter name for one parameter, you have to use names for all of them.
- If you don't use parameter names, the member set values must appear in the parameter list in the exact order that WPWin expects them. This appendix, and the WPWPxx.WCD file, show the proper order of parameters.
- In many cases, you can omit one or more of the parameters in a command. WPWin uses default values for the missing parameters. However, if you skip a parameter you'll need to use parameter names so that WPWin will know which parameters you are setting.

A good rule of thumb to follow about parameter names: If in doubt, include the name. Of course, you are always free to experiment; WPWin will tell you if something is amiss.

Using Embedded Codes

Some of the product function commands, such as **SearchText** and **SearchReplace**, expect a special kind of string parameter: an embedded code. These codes are generated when WPWin is recording a macro. The code identifies a WordPerfect format or function.

If you need to use a product command that requires an embedded code, you'll need to record a macro that produces it. This is most easily done by manually repeating the critical steps of the macro you are writing while recording a macro. After the macro is recorded, open it, and copy the embedded code.

AboutDlg

Function: Calls the About WordPerfect dialog.

Equivalent Command: Help-About WordPerfect

Syntax
AboutDlg()

Parameters
None

Advance

Function: Places text at a certain place on the page, or at a specific distance from the insertion point.

Equivalent Command: Advance dialog options;
Layout-Advance

Syntax
Advance (Where:{
Up!=0;
Down!=1;
ToLine!=2;
Left!=3;
Right!=4;
ToPosition!=5
};
Amount:wpunits
)

Parameters
Where: Sets direction for advance function.

Amount: When used with **ToLine!** and **ToPosition!**, sets absolute position of Advance code. Text will appear at that spot on the page. When used with the remaining Where: parameters, sets position relative to the insertion point.

AdvanceDlg

Function: Calls the Advance dialog box.

Equivalent Command: Layout-Advance

Syntax
AdvanceDlg()

Parameters
None

AppMaximize

Function: Maximizes the WordPerfect program window.

Equivalent Command: (Control Menu Box)-Maximize

Syntax
AppMaximize()

Parameters
None

AppMinimize

Function: Minimizes the WordPerfect program window.

Equivalent Command: (Control Menu Box)-Minimize

Syntax AppMinimize()

Parameters
None

AppMove

Function: Moves the WordPerfect application window. As this command has no positioning parameters, the macro will pause and allow you to adjust the position manually. The macro resumes when you press **[Enter]** or click the mouse.

Equivalent Command: (Control Menu Box)-Move

Syntax
AppMove()

Parameters
None

AppRestore

Function: Restores the WordPerfect application window to its preset size (the size before maximizing or minimizing).

Equivalent Command: (Control Menu Box)-Restore

Syntax
AppRestore()

Parameters
None

AppSize

Function: Resizes the WordPerfect application window. As this command has no sizing parameters, the macro will pause and allow you to adjust the size manually. The macro resumes when you press **[Enter]** or click the mouse.

Equivalent Command: (Control Menu Box)-Size

Syntax
AppSize()

Parameters
None

BlockProtect

Function: Turns block protection on for selected text. Block-protected text will not break at a new page.

Equivalent Command: Layout-Page-Block Protect

Syntax
BlockProtect()

Parameters
None

BoxEditCaption

Function: Creates or edits a caption for the currently selected graphic box.

Equivalent Command: Graphic-**{any}**-Caption

Syntax
BoxEditCaption()

Parameters None

Graphic box must be selected first.

BoxEditor

Function: Opens one of three box editors: Figure, Text, or Equation.

Equivalent Command: Graphic box editor;
Graphic-**{any}**-Edit

Syntax

```
BoxEditor(
        Type:{
                Figure!=0;
                Table!=1;
                Text!=2;
                User!=3;
                Equation!=4
        };
        Editor:{
                Figure!=0;
                Text!=1;
                Equation!=2
        }
)
```

Parameters

Type: Specifies the graphic type.

Editor: Specifies the graphic editor to use. If no Editor parameter is provided, BoxEditor opens the appropriate editor type depending on the currently selected box (Figure Editor for a figure graphic, etc.). WordPerfect overrides the Editor parameter if it does not match the contents of the selected box.

BoxFigureEdit

Function:
Sets options within the Figure Editor (including scaling, mirroring, and outlining).

Equivalent Command: Various; from within graphics editor.

Syntax

```
BoxFigureEdit(
        Filename:string;
        ScaleX:number;
        ScaleY:number;
        Rotate:number;
        MoveX:wpunits;
        MoveY:wpunits;
        Mirror:{
                No!=0;
                Yes!=1
        };
        Invert:{
                No!=0;
                Yes!=1
        };
        BlackAndWhite:{
                No!=0;
                Yes!=1
        };
        Outline;{
                No!=0;
                Yes!=1
        };
        GraphOnDisk:{
                No!=0;
                Yes!=1
        };
        SaveFilename:string
)
```

Parameters
Filename: Optional; indicates filename for graphic if one is to be retrieved for editing. This argument is a string so be sure to enclose it in quotes.

ScaleX: Scales width of the figure. Value from 1 to 999.

ScaleY: Scales height of the figure. Value from 1 to 999.

Rotate: Rotates the figure, in degrees. Value from 0 to 359. The direction of rotation is clockwise.

MoveX: Moves figure horizontally. Value is in wpunits. Positive value moves figure to the right; negative value moves figure to the left.

MoveY: Moves figure vertically. Value is in wpunits. Positive value moves figure up; negative value moves figure down.

Mirror: Sets mirror (flip left/right) on or off.

Invert: Sets color inversion (black becomes white, etc.).

BlackAndWhite: Sets display of graphic in black and white.

Outline: Sets outline display for figure on or off.

GraphOnDisk: Indicates whether figure will be stored on disk for use in the file.

SaveFilename: If GraphOnDisk is set to Yes (1), specifies disk filename for graphic. This argument is a string, so be sure to enclose it in quotes.

Figure Editor must already be active.

BoxNewNumber

Function: Starts a new numbering sequence for subsequent graphic boxes. The new numbering affects only the graphic box type specified.

Equivalent Command: Macro only.

Syntax

```
BoxNewNumber(
        Type:{
                Figure!=0;
                Table!=1;
                Text!=2;
                User!=3;
                Equation!=4
        }
        BoxNumber:string
)
```

Parameters

Type: Specifies type of graphic box for new number. The box number for only the specified graphic type is affected.

BoxNumber: New number of box. You can use any valid WordPerfect numbering sequence, including roman (upper- or lower-case), and arabic numerals. This argument is a string, so enclose it in quotes, such as "10".

BoxOptions

Function: Sets options (including border, spacing, numbering, and shading) for graphic boxes. Options for only the box type specified are affected.

Equivalent Command: Set options in graphic options dialog box
Graphic-**{any}**-Options

Syntax

```
BoxOptions(
    Type:{
        Figure!=0;
        Table!=1;
        Text!=2;
        User!=3;
        Equation!=4
    };
    LeftBorder:{
        None!=0;
        Single!=1;
        Double!=2;
        Dashed!=3;
        Dotted!=4;
        Thick!=5;
        ExtraThick!=6
    };
    RightBorder:{
        <same as LeftBorder>
    };
    TopBorder:{
        <same as LeftBorder>
    };
    BottomBorder:{
        <same as LeftBorder>
    };
    OutLeftSpace:wpunits;
    OutRightSpace:wpunits;
    OutTopSpace:wpunits;
    OutBottomSpace:wpunits;
    InLeftSpace:wpunits;
    InRightSpace:wpunits;
    InTopSpace:wpunits;
    InBottomSpace:wpunits;
    FirstLevelNumber:{
        Off!=0;
        Numbers!=1;
        Letters!=2;
        Roman!=3
    };
    SecondLevelNumber:{
        <same as FirstLevelNumber>
    };
    CaptionPosition:{
```

```
            BelowOutside!=0;
            AboveOutside!=1;
            BelowInside!=2;
            AboveInside!=3;
            Below!=4;
            Above!=5;
            Left!=6;
            Right!=7
    };
    CaptionStyle:string;
    MinimumOffset:wpunits;
    GrayShade:number
```

Parameters

Type: Specifies box type.

LeftBorder: Specifies border frame option, such as no frame, double frame, etc.

RightBorder: Same as LeftBorder.

TopBorder: Same as LeftBorder.

BottomBorder: Same as LeftBorder.

OutLeftSpace: Specifies distance, in wpunits, between normal text and box frame.

OutRightSpace: Same as OutLeftSpace.

OutTopSpace: Same as OutLeftSpace.

OutBottomSpace: Same as OutLeftSpace.

InLeftSpace: Specifies distance, in wpunits, between box frame an contents of box.

InRightSpace: Same as InLeftSpace.

InTopSpace: Same as InLeftSpace.

InBottomSpace: Same as InLeftSpace.

FirstLevelNumber, SecondLevelNumber: Specifies numbering style for first (or second) level caption.

CaptionPosition: Specifies caption position.

CaptionStyle: Running text for the caption. Parameter is string; enclose it in quotes.

MinimumOffset: Specifies distance between top of the paragraph and the graphic.

GrayShade: Specifies shading inside the graphic box, if any. Value: 0 to 100 (0 is no shading; 100 is all black).

BoxPosition

Function: Sets position for graphic boxes. Position affects only the box type specified.

Equivalent Command: Set options in graphic placement dialog box;
Graphic-**{any}**-Placement

Syntax

```
BoxPlacement(
    NewType:{
        Figure!=0;
        Table!=1;
        Text!=2;
        User!=3;
        Equation!=4
    };
    Anchor:{
        Paragraph!=0;
        Page!=1;
        Character!=2
    };
    SkipPages:number;
    AutoMode:{
        AutoBoth!=0;
        AutoWidth!=1;
        AutoHeight!=2;
        SetBoth!=3
    };
    Width:number;
    Height:number;
    VerticalType:{
        FullPage!=0;
        Top!=1;
        Center!=2;
        Bottom!=3;
        Baseline!=4;
        Specify!=5
    };
    VerticalPosition:number;
    HorizontalType:{
        Left!=0;
        Right!=1;
        Center!=2;
        Full!=3;
        ColumnLeft!=4;
        ColumnRight!=5;
        ColumnCenter!=6;
        ColumnFull=7;
        Specify!=8
    };
    HorizontalPosition:number;
    ColumnStart:number;
    ColumnEnd:number;
```

```
        WrapText:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

NewType: Changes the selected box to the specified type. Omit argument if box type is to remain the same.

Anchor: Specifies anchor type.

SkipPages: Specifies number of pages to skip before the graphic appears in the document. Use this argument only when Anchor is set to **Page!**.

AutoMode: Specifies sizing mode of the graphic (automatic, set height, set width, or set both).

Width: Specifies width of graphic, in wpunits.

Height: Specifies height of the graphic, in wpunits.

VerticalType: Sets vertical alignment/position for box.

VerticalPosition: Specifies exact vertical position of graphic, in wpunits. Not used unless VerticalType: is set to **Specify!**

HorizontalType: Sets horizontal alignment/position for box.

HorizontalPosition: Specifies exact horizontal position of graphic, in wpunits. Not used unless HorizontalType: is set to **Specify!**.

ColumnStart: Specifies starting (leftmost) column when box spans across two or more columns. Value: positive whole number only.

ColumnEnd: Specifies ending (rightmost) column when box spans across two or more columns. Value: positive whole number only.

WrapText: Indicates whether document text is to wrap around box. Value: -1 for TRUE (wrap text); 0 for FALSE (do not wrap text).

BoxRetrieve

Function: Retrieves a previously saved graphic at the insertion point.

Equivalent Command: Graphics-Figure-Retrieve

Syntax

```
BoxRetrieve(
        Type:{
                Figure!=0;
                Table!=1;
                Text!=2;
                User!=3;
                Equation!=4
        };
        Filename:string
}
```

Parameters

Type: Specifies the type of figure that is retrieved.

Filename: Specifies the filename of the figure. Enclose filename (with path, if necessary) in quotes.

BoxSelect

Function: Selects a box following a specified type and number. The appropriate box must already exist.

Equivalent Command: Macro only.

Syntax

```
BoxSelect(
        Type:{
                Figure!=0;
                Table!=1;
                Text!=2;
                User!=3;
                Equation!=4
         };
        BoxNumber:string
)
```

Parameters

Type: Specifies a box type for selection.

BoxNumber: Box number to select. You can use any valid WordPerfect numbering sequence, including roman (upper- or lower-case), and arabic numerals. Enclose the BoxNumber: argument in quotation marks, as in BoxNumber:"10". Omitting the BoxNumber: parameter forces WordPerfect to select the next box number (for the specified type).

ButtonBarEditDlg

Function: Calls the Button Bar edit dialog.

Equivalent Command: View-Button Bar Setup-Edit

Syntax
ButtonBarEditDlg()

Parameters
None

ButtonBarNewDlg

Function: Calls the Edit Button Bar dialog.

Equivalent Command: View-Button Bar Setup-New

Syntax
ButtonBarNewDlg()

Parameters
None

ButtonBarOptions

Function: Sets options for current Button Bar.

Equivalent Command: Settings in Button Bar options dialog;
Views-Button Bar Setup-Options

Syntax

```
ButtonBarOptions(
	Style:{
		PictureOnly!=0;
		TextOnly!=1;
		PictureAndText!=2
	};
	Location:{
		BBarOnTop!=0;
		BBarOnLeft!=1;
		BBarOnRight!=2;
		BBarOnBottom!=3
	}
)
```

Parameters

Style: Specifies type of button (picture only, text only, or text and picture).

Location: Specifies location for the Button Bar.

ButtonBarOptionsDlg

Function: Calls the Button Bar Options dialog.

Equivalent Command: View-Button Bar Setup-Options

Syntax
ButtonBarOptionsDlg()

Parameters
None

ButtonBarSave

Function: Saves Button Bar changes.

Equivalent Command: Save Button Bar from Button Bar Save As dialog;
View-Button Bar Setup-Save As

Syntax
ButtonBarSave(
FileName:string
)

Parameters
FileName: Filename for Button Bar. Enclose the filename in quotes; include a path (if necessary).

ButtonBarSaveDlg

Function: Calls Button Bar Save As dialog.

Equivalent Command: View-Button Bar Setup-Save As

Syntax
ButtonBarSaveDlg()

Parameters
None

ButtonBarSelect

Function: Selects a specified Button Bar.

Equivalent Command: Select Button Bar from Button Bar open dialog;
View-Button Bar Setup-Select

Syntax
ButtonBarSelect(
FileName:string
)

Parameters
FileName: Filename for Button Bar. Enclose the filename in quotes; include a path (if necessary).

ButtonBarSelectDlg

Function: Calls the Select Button Bar dialog.

Equivalent Command: View-Button Bar Setup-Select

Syntax
ButtonBarSelectDlg()

Parameters
None

ButtonBarShow

Function: Sets Button Bar display on or off. ButtonBarShow ignores the current setting and forces the display on or off, as specified.

Equivalent Command: View-Button Bar

Syntax

```
ButtonBarShow(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Forces Button Bar display on or off (regardless of current setting).

CaptionBoxNumber

Function: Inserts the current caption box number into the caption editor at the insertion point. The caption editor should already be active.

Equivalent Command: (in caption editor) [Number]

Syntax
CaptionBoxNumber()

Parameters
None

CaseToLower

Function: Converts selected text to lower-case. Text must be selected before the command is used.

Equivalent Command: Edit-Convert Case-Lower

Syntax
CaseToLower()

Parameters
None

CaseToUpper

Function: Converts selected text to upper-case. Text must be selected before the command is used.

Equivalent Command: Edit-Convert Case-Upper

Syntax
CaseToUpper()

Parameters
None

Close

Function: Closes the current document screen. Can be used to close main Edit documents as well as subdocuments (footnotes, headers, footers, etc.).

Equivalent Command: File-Close

Syntax

```
Close(
        Save:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

None

CloseNoSave

Function: Discards the active document without saving it or closing its window (i.e., the document window is cleared, but remains open).

Equivalent Command: Macro only.

Syntax

```
CloseNoSave(
        Verify:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Save: Specifies whether changes to the file should be saved before closing.

ColumnDefine

Function: Inserts column definition code.

Equivalent Command: Column Define dialog options;
Layout-Columns-Define

Syntax

```
ColumnDefine(
        NumberOfColumns:number;
        Type:{
                Newspaper!=0;
                Parallel!=1;
                ParallelWithBlockProtect!=2
        };
        ColumnsOn:{
                No!=0;
                Yes!=1
        };
        LeftMargin1:wpunits;
        RightMargin1:wpunits;
                ...
        LeftMargin24:wpunits;
        RightMargin24:wpunits
)
```

Parameters

NumberOfColumns: Specifies the number of columns. Value: 1 to 24.

Type: Specifies the column type, either newspaper, parallel, or parallel with block protect.

ColumnsOn: Specifies if columns are turned on automatically after column definition (no need to turn columns on with ColumnOn command).

LeftMargin# (1 through 24): Specifies left margin position for numbered columns.

RightMargin# (1 through 24): Specifies right margin position for numbered columns.

- WordPerfect supports up to 24 columns. The program will ignore additional definitions.
- Be certain that margin settings increase in value for each LeftMargin#: and RightMargin#: setting, or the margins may overlap.
- You can use any valid wpunits measurement for LeftMargin#: and RightMargin#:.

ColumnDefineDlg

Function: Calls the Column Define dialog.

Equivalent Command: Layout-Columns-Define

Syntax
ColumnDefineDlg()

Parameters
None

ColumnDefineEven

Function: Inserts column definition code for evenly sized columns (column margins not required; WordPerfect calculates column dimensions for you).

Equivalent Command: Layout-Columns-Define

Syntax

```
ColumnDefineEven(
        NumberOfColumns:number;
        Type:{
                Newspaper!=0;
                Parallel!=1;
                ParallelWithBlockProtect!=2
        };
        ColumnsOn:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

NumberOfColumns: Specifies the number of columns. Value: 1 to 24.

Type: Specifies the column type, either newspaper, parallel, or parallel with block protect.

ColumnsOn: Specifies if columns are turned on automatically after column definition (no need to turn columns on with ColumnsOn command).

ColumnsOff

Function: Turns columns off.

Equivalent Command: Layout-Columns-Columns Off

Syntax
ColumnsOff()

Parameters None

Columns must already be defined.

ColumnsOn

Function: Turns columns on.

Equivalent Command: Layout-Columns-Columns On

Syntax
ColumnsOn()

Parameters
None

Columns must already be defined.

CommentConvertToText

Function: Converts a comment to document text.

Equivalent Command: Tools-Comment-Convert to Text

Syntax
CommentConvertToText()

Parameters
None

WordPerfect automatically selects the preceding comment for conversion. If there is no preceding comment, WordPerfect chooses the next comment in text it encounters.

CommentCreate

Function: Creates or edits a comment.

Equivalent Command: Tools-Comment-Create

Syntax

```
CommentCreate(
        CommentText:string;
        Replace:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

CommentText: Specifies the text of the comment. Enclose the text in quotes.

Replace: When editing an existing comment, specifies whether CommentText replaces old text; or is added to the existing text at the current insertion point.

CommentCreateDlg

Function: Calls the Comment Create dialog.

Equivalent Command: Tools-Comment-Create

Syntax
CommentCreateDlg()

Parameters
None

CommentEditDlg

Function: Calls the Comment Edit dialog.

Equivalent Command: Tools-Comment-Edit

Syntax CommentEditDlg()

Parameters
None

- To effectively use **CommentEditDlg**, position the cursor immediately after the comment you want to edit.
- If WordPerfect can't find a comment preceding the cursor, it will edit the first comment box in the document.
- If no comments exist in the document, WordPerfect will open an empty comment.

CommentShow

Function: Turns comment display on or off. **CommentShow** forces the display either on or off, as specified, regardless of the current setting of the View-Comments command.

Equivalent Command: View-Comments

Syntax

```
CommentShow(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies whether comments are displayed

ConditionalEOP

Function: Inserts a Conditional End of Page command, which keeps a specified number of lines of a paragraph together when a page break occurs.

Equivalent Command: Layout-Page-Conditional End of Page

Syntax
ConditionalEOP(
KeepTogether:number
)

Parameters
KeepTogether: Specifies the number of lines to keep together when a page break appears. Value: positive whole number only (indicating the number of lines to keep together).

ConditionalEOPDlg

Function: Calls the Conditional End of Page dialog.

Equivalent Command: Layout-Page-Conditional End of Page

Syntax
ConditionalEOPDlg()

Parameters
None

ConvertClipboardDlg

May '92

Function: Calls the Convert Windows Picture dialog box.

Equivalent Command: File-Preference-Graphics Import

Syntax
ConvertClipboardDlg()

Parameters
None

ConvertClipboardPicture 

Function: Specifies the format for graphics imported via the Windows Clipboard. Choices are convert picture to Windows metafile format, WordPerfect WPG format, or both

Equivalent Command: File-Preference-Graphics Import

Syntax

```
ConvertClipboardPicture(
        ConvertTo:{
                Metafile!=0;
                WPG!=1;
                Both!=2
        }
)
```

Parameters

ConvertTo: Specifies the graphics format.

CrossRefMark

Function: Creates a cross reference mark, consisting of a target name, a reference name, or both.

Equivalent Command: Tools-Mark Text-Cross-Reference

Syntax

```
CrossRefMark(
        TargetName:string;
        MarkType:{
                TargetOnly!=0;
                RefOnly!=1;
                BothMode!=2
        };
        RefType:{
                PageNumber!=0;
                ParagraphOutline!=1;
                FootnoteNumber!=2;
                EndnoteNumber!=3;
                FigureNumber!=4;
                TableBoxNumber!=5;
                TextBoxNumber!=6;
                UserBoxNumber!=7;
                EquationBoxNumber!=8
        }
)
```

Parameters

TargetName: Specifies the name of the cross-reference. This parameter is a string, so enclose it in quotes.

MarkType: Specifies how the cross-reference is marked: for the target only, for the reference only, or for both.

RefType: Specifies the object of the reference, such as the page number or figure number.

- If MarkType is **TargetOnly!**, the Target code is placed at the current insertion point.
- If MarkType is **RefOnly!**, the Ref code is placed at the current insertion point.
- If MarkType is **BothMode!**, and the macro contains no subsequent insertion point movement commands, the Ref code is placed at the current insertion point. WordPerfect prompts you to move the insertion point to the target and press **[Enter]**.
- The macro can contain insertion point movement commands after the **CrossRefMark** to automatically position the Target code. Include a **HardReturn** command to place the code at the current insertion point.

CrossRefMarkDlg

Function: Calls the Mark Cross-Reference dialog box.

Equivalent Command: Tools-Mark Text-Cross-Reference

Syntax
CrossRefMarkDlg()

Parameters
None

DateCode

Function: Inserts date code.

Equivalent Command: Tools-Date-Code

Syntax
DateCode()

Parameters
None

DateFormat

Function: Specifies a new date format for the document and for the remainder of the current WordPerfect session (does not permanently change the default date format).

Equivalent Command: Tools-Date-Format

Syntax
DateFormat(
DefinitionString:string
)

Parameters
DefinitionString: Specifies the date format. This parameter is a string so enclose it in quotes. See the PrefDateFormat command for an explanation of how to specify the date format.

DateFormatDlg

Function: Calls the Date Format dialog.

Equivalent Command: Tools-Date-Format

Syntax
DateDefineDlg()

Parameters
None

DateText

Function: Inserts date text.

Equivalent Command: Tools-Date-Text

Syntax
DateText()

Parameters
None

DDECreateLink

Function: Creates a new dynamic data exchange (DDE) link.

Equivalent Command: Create Link dialog options;
Edit-Link-Create

Syntax

```
DDECreateLink(
        LinkName:string;
        Source:string;
         UpdateMode:{
                Manual!=0;
                Automatic!=1
        };
        StoragePreference:{
                Graphics!=0;
                Text!=1
        }
)
```

Parameters

LinkName: Specifies a unique name for the link, for references purposes. Enclose the string in quotes.

Source: Specifies the source of the DDE link. The parameter follows this format: **Application|Document|Range** (**Range** is optional; if omitted, the entire document is linked). Enclose the string in quotes. Use the "pipe" symbol to separate the **Appolication**, **Document**, and **Range** parameters.

UpdateMode: Indicates whether link updates are manual or automatic.

StoragePreference: Indicates storage preference; graphics or text.

- DDE links cannot be established unless the source application is already running, and unless the source document is already loaded.
- WordPerfect and Windows may become sluggish during **Automatic!** updates. If possible, set to **Manual!**.
- Performance is degraded when StoragePreference: is set to **Graphics!**. If the link passes only text, set StoragePreference: to **Text!**.

DDECreateLinkDlg

Function: Calls Create Link dialog.

Equivalent Command: Edit-Link-Create

Syntax
DDECreateLinkDlg()

Parameters
None

DDEDeleteLink

Function: Deletes an active dynamic data exchange (DDE) link.

Equivalent Command: Delete DDE Link dialog options;
Edit-Link-Delete

Syntax
DDEDeleteLink(
LinkName:string;
...
LinkName20:string
)

Parameters
LinkName: Deletes the indicated DDE link, by name (not source). You can delete up to 20 links, as this is the total number of DDE links supported by WordPerfect. Indicate the name of each link you wish to delete, followed by a semicolon (;).

DDEDeleteLinkDlg

Function: Calls the Delete DDE Link dialog.

Equivalent Command: Edit-Link-Delete

Syntax
DDEDeleteLinkDlg()

Parameters
None

DDEEditLink

Function: Changes the properties of an existing dynamic data exchange (DDE) link. This includes changing the link name, the source, and the update mode.

Equivalent Command: Edit-Link-Edit

Syntax

```
DDEEditLink(
        LinkName:string;
        NewName:string;
        NewSource:string;
        NewUpdateMode:{
                Manual!=0;
                Automatic!=1
        }
)
```

Parameters

LinkName: Identifies the existing DDE link you want to change. Value is a string; enclose the argument in quotes.

NewName: Specifies the new name for the DDE link. Value is a string; enclose the argument in quotes.

NewSource: Specifies the new source for the DDE link. The parameter follows this format: **Application|Document|Range** (**Range** is optional; if omitted the entire document is linked). Value is a string; enclose the argument in quotes. Use the "pipe" symbol to separate the **Appolication**, **Document**, and **Range** parameters.

NewUpdateMode: Indicates whether link updates are manual or automatic.

See DDECreateLink for important notes on using DDE links.

DEEditLinkDlg

Function: Calls the DDE Edit Link dialog.

Equivalent Command: Edit-Link-Edit

Syntax
DDEEditLinkDlg()

Parameters
None

DDEPasteLink

Function: Establishes a link between a WordPerfect document and an application via the Windows Clipboard. The source application must first copy data to the Clipboard. The **DDEPasteLink** command then retrieves this data, and makes a DDE link out of it. There is no need to specify an application, filename, or range for the DDE link, as all this information is provided in the Clipboard connection.

Equivalent Command: Edit-Link-Paste

Syntax
DDEPasteLink()

Parameters
None

DDEUpdateLink

Function: Updates an already established and active link.

Equivalent Command: Edit-Link-Update

Syntax

```
DDEUpdateLink(
        LinkName:string;
        LinkName2:string;
        ...
        LinkName20:string
)
```

Parameters

DDEUpdateLink: Specifies the link to update (up to 20 active DDE links are supported by WordPerfect). Indicate the name of each link you wish to update, followed by a semicolon (;).

DDEUpdateLinkDlg

Function: Calls the Update DDE Link dialog box.

Equivalent Command: Edit-Link-Update

Syntax
DDEUpdateLinkDlg()

Parameters
None

DeleteCharNext

Function: Deletes the next character in text.

Equivalent Command: **[Delete]**

Syntax
DeleteCharNext()

Parameters
None

DeleteCharPrevious

Function: Deletes the previous character in text.

Equivalent Command: **[Bksp]** (or) **[Shift+Bksp]**)

Syntax
DeleteCharPrevious()

Parameters
None

DeleteEOL

Function: Deletes all text and codes from the insertion point to the end of the current line.

Equivalent Command: **[Ctrl+Del]**

Syntax
DeleteEOL()

Parameters
None

DeleteEOP

Function: Deletes all text and codes from the insertion point to the end of the current page.

Equivalent Command: **[Ctrl+Shift+Del]**

Syntax
DeleteEOP()

Parameters
None

DeleteToBeginningOfWord

Function: Deletes from the insertion point to the beginning of the current word.

Equivalent Command: None

Syntax
DeleteToBeginningOfWord()

Parameters
None

DeleteToEndOfWord

Function: Deletes from the insertion point to the end of the current word.

Equivalent Command: None

Syntax
DeleteToEndOfWord()

Parameters
None

DeleteWord

Function: Deletes one or more words beginning at the insertion point.

Equivalent Command: **[Ctrl+Bksp]**

Syntax
DeleteWord()

Parameters
None

Display

Function: Turns text writing on or off during macro execution.

Equivalent Command: Macro only.

Syntax

```
Display (
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies whether text is displayed as it is typed by the macro (usually as a result of the **Type** command). When State is off, text inserted during macro execution is not shown on the screen. When is on, text is shown as it is being inserted.

- The default setting (without issuing a **Display** command in the macro) is **Display** (Off!).
- The macro will run faster when display is turned off.
- Text inserted by the macro will display when the macro finishes.
- If you are inserting only small pieces of text (only a few words), execution speed will not be noticeably degraded if the display is kept on.

DisplayFontAdjustment

Function: Specifies the size of font characters as a percentage of their normal size. Works only when WordPerfect is not in Draft mode.

Equivalent Command: Layout-Document-Display Pitch

Syntax

```
DisplayFontAdjustment(
        Mode:{
                Auto!=0;
                Manual!=1
        };
        PercentOfNormal:number
        }
)
```

Parameters

Mode: Specifies if font display is automatically adjusted by WordPerfect or adjusted manually.

PercentOfNormal: When Mode is set to **Manual!**, specifies size of font, in percentage. Valid values: Whole numbers from 5 to 250.

DisplayPitch

Function: Sets the horizontal space between characters on screen when WPWin is set to Draft mode.

Equivalent Command: Display Pitch dialog options;
Layout-Document-Display Pitch

Syntax

```
DisplayPitch(
        Type:{
                Auto!=0;
                 Manual!=1
        };
        Pitch:wpunits
)
```

Parameters

Type: Specifies if font pitch is automatically adjusted by WordPerfect or is adjusted manually.

Pitch: When Type: is set to **Manual!**, indicates pitch between characters. Value is in wpunits, in a range from 0.025" to 0.500".

DisplayPitchDlg

Function: Calls the Display Pitch dialog box.

Equivalent Command: Layout-Document-Display Pitch

Syntax
DisplayPitchDlg()

Parameters
None

DocCompare

Function: Compares the active document against a specified document that has been previously saved. Differences, if any, are noted by redlining (added text) and strikeout (deleted text). In addition, text that has been moved to a different location in the document is noted.

Equivalent Command: Tools-Document Compare-Add Markings

Syntax
DocCompare(
ComparisonFile:string
)

Parameters
ComparisonFile: Specifies the previously saved document to compare against the active document. The argument is a string, so enclose it in quotes.

DocCompareAddMarkingsDlg

Function: Calls the Add Markings dialog.

Equivalent Command: Tools-Document Compare-Add Markings

Syntax
DocCompareAddMakingsDlg()

Parameters
None

DocCompareRemove

Function: Removes redline and strikeout text after a document has been compared using the DocCompare command.

Equivalent Command: Tools-Document Compare-Remove Markings

Syntax

```
DocCompareRemove(
        KeepRedline:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

KeepRedline: Specifies if you want to keep redlined (added) text.

DocCompareRemoveMarkingsDlg

Function: Calls the Remove Markings dialog.

Equivalent Command: Tools-Document Compare-Remove Markings

Syntax
DocCompareRemoveMakingsDlg()

Parameters
None

DocInitialCodes

Function: Opens the Initial Codes window for setting initial codes for the entire document.

Equivalent Command: Layout-Document-Initial Codes

Syntax
DocInitialCodes()

Parameters
None

Actions following this command apply to the Document Initial Codes window. When you are done, Close to return to the main Edit window.

DocInitialFont

Function: Sets the default font for the active document.

Equivalent Command: Document Initial Font dialog options;
Layout-Document-Initial Font

Syntax
DocInitialFont(
Name:string;
Size:number
)

Parameters
Name: Specifies the name of the font to use. Enclose the name in quotes.

Size: Specifies the point size of the font. You can use any number, with or without a decimal point and decimal places.

- Name should correspond to the actual name of a font for the currently selected printer. If you're not sure of the correct name, check first.
- Not all fonts are available in all sizes. Verify first that a font is available in a certain size before specifying it in the **DocInitialFont** command.

DocInitialFontDlg

Function: Calls the Document Initial Font dialog.

Equivalent Command: Layout-Document-Initial Font

Syntax
DocInitialFontDlg()

Parameters
None

DocMaximize

Function: Maximizes the current document.

Equivalent Command: (Document Control Menu Box)-Maximize

Syntax
DocMaximize()

Parameters
None

DocMinimize

Function: Minimizes the current document.

Equivalent Command: (Document Control Menu Box)-Minimize

Syntax
DocMinimize()

Parameters
None

DocMove

Function: Moves the current document. As this command has no positioning parameters, the macro will pause and allow you to adjust the position manually. The macro resumes when you press **[Enter]** or click the mouse.

Equivalent Command: (Document Control Menu Box)-Move

Syntax
DocMove()

Parameters
None

DocNext

Function: Moves the active document to the back and makes the next open document (if any) in the stack active. The command is ignored if there are no other documents.

Equivalent Command: (Document Control Menu Box)-Next

Syntax
DocNext()

Parameters
None

DocPrevious

Function: Moves the last document (if any) in the stack to the front. This document becomes active. The **DocPrevious** command is active only in Windows 3.1 or later; when used in Windows 3.0 it has the same effect as DocNext. The command is ignored if there are no other documents.

Equivalent Command: Macro only.

Syntax
DocPrevious()

Parameters
None

DocRedlineMethod

Function: Specifies how redlined text will appear.

Equivalent Command: Redline Method dialog options;
Layout-Document-Redline Method

Syntax

```
DocRedlineMethod(
        Method:{
                PrinterDependent!=0;
                Left!=1;
                Alternating!=2
        };
        Character:string
)
```

Parameters

Method: Specifies how redline markings will appear.

Character: When **Left!** or **Alternating!** is used with the Method: parameter, specifies the character placed in the margin to denote changed text.

DocRedlineMethodDlg

Function: Calls the Redline Method dialog.

Equivalent Command: Layout-Document-Redline Method

Syntax
DocRedlineMethodDlg()

Parameters
None

DocRestore

Function: Restores the current document window to its preset size (the size before maximizing or minimizing).

Equivalent Command: (Document Control Menu Box)-Restore

Syntax
DocRestore()

Parameters
None

DocSize

Function: Resizes the current document window. As this command has no sizing parameters, the macro will pause and allow you to adjust the size manually. The macro resumes when you press **[Enter]** or click the mouse.

Equivalent Command: (Document Control Menu Box)-Size

Syntax
DocSize()

Parameters
None

DocSummaryDefine

Function: Specifies the text to be entered in the various fields of the Document Summary dialog.

Equivalent Command: Document Summary dialog options;
Layout-Document-Summary

Syntax

```
DocSummaryDefine(
    Name:string;
    Type:string;
    Date:string;
    Author:string;
    Typist:string;
    Subject:string;
    Account:string;
    KeyWords:string;
    Abstract:string
)
```

Parameters

All of the following are strings, limited in length (specified in bytes) as specified. All strings should be enclosed in quotes.

Name: Specifies text for the long document name field. Maximum length: 68.

Type: Specifies text for the document type field. Maximum length: 21.

Date: Specifies text for the date of creation or revision. Maximum length: 26.

Author: Specifies text for the author. Maximum length: 61.

Typist: Specifies text for the typist. Maximum length: 61.

Subject: Specifies text for the subject. Maximum length: 161.

Account: Specifies text for the account. Maximum length: 161.

KeyWords: Specifies text for the key words. Maximum length: 161.

Abstract: Specifies text for the abstract. Maximum length: 781.

DocSummaryDelete

Function: Deletes all document summary information from the current document.

Equivalent Command: Layout-Document-Summary-[Delete]

Syntax
DocSummaryDelete()

Parameters
None

Creation date information is always set by WordPerfect and cannot be deleted.

DocSummaryDlg

Function: Displays the Document Summary dialog box.

Equivalent Command: Layout-Document-Summary

Syntax
DocSummaryDlg()

Parameters
None

DocSummaryExtract

Function: Extracts text from the current document and places it into the appropriate fields of the document summary.

Equivalent Command: Layout-Document-Summary-[Extract]
Syntax
DocSummaryExtract()

Parameters
None

- The Author and Typist fields are filled with the same text used in the document summary of the previously edited document.
- If the document contains a subject search text (normally RE:), WordPerfect will copy the first 150 characters following this text into the Subject field. You can specify another Subject Search Text in the Document Summary Preferences dialog (File-Preferences-Document Summary).
- The first 400 characters of the document are placed into the Abstract field.

DocSummaryGetData

Function: Retrieves the contents of one or more document summary fields and assigns the contents to a variable.

Equivalent Command: Macro only.

Syntax

```
DocSummaryGetData(
        Name:variable;
        Type:variable;
        Date:variable;
        Author:variable;
        Typist:variable;
        Subject:variable;
        Account:variable;
        KeyWords:variable;
        Abstract:variable
)
```

Parameters

All of the following are variable names.

Name: Specifies the variable for the long document name field.

Type: Specifies the variable for the document type field.

Date: Specifies the variable for the date of creation or revision field.

Author: Specifies the variable for the author field.

Typist: Specifies the variable for the typist field.

Subject: Specifies the variable for the subject field (the Subject Search Text, typically RE:, is not included).

Account: Specifies the variable for the account field.

KeyWords: Specifies the variable for the key words field.

Abstract: Specifies the variable for the abstract field.

DocSummaryPrint

Function: Prints the document summary of the current document.

Equivalent Command: Layout-Document-Summary-[Print]

Syntax
DocSummaryPrint()

Parameters
None

DocSummarySaveAs

Function: Copies the specified document summary fields to another document (the text for the fields is provided in the arguments to the **DocSummarySaveAs** command, not the current contents of the Document Summary dialog box) . The destination file then contains a listing of the document summary fields. If you specify an existing document, WordPerfect will ask if you want to Overwrite the existing document, or Append to it. When Appending, WordPerfect adds the document summary to the end of the document.

Equivalent Command: None

Syntax

```
DocSummarySaveAs(
        Filename:string;
        DocName:string;
        DocType:string;
        CreationDate:string;
        Author:string;
        Typist:string;
        Subject:string;
        Account:string;
        KeyWords:string;
        Abstract:string
)
```

Parameters

All the arguments are strings, and should be enclosed in quotes. Lengths are specified in bytes.

Filename: Specifies the name of the file to save the document summary text to.

DocName: Specifies text for the long document name field. Maximum length: 68.

DocType: Specifies text for the document type field. Maximum length: 21.

CreationDate: Specifies text for the date of creation or revision. Maximum length: 26.

Author: Specifies text for the author. Maximum length: 61.

Typist: Specifies text for the typist. Maximum length: 61.

Subject: Specifies text for the subject. Maximum length: 161.

Account: Specifies text for the account. Maximum length: 161.

KeyWords: Specifies text for the key words. Maximum length: 161.

Abstract: Specifies text for the abstract. Maximum length: 781.

DraftMode

Function: Turns Draft mode on or off. When used without a parameter, **DraftMode** forces Draft mode on or off, regardless of the current setting.

Equivalent Command: View-Draft Mode

Syntax

```
DraftMode(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Indicates whether draft mode is turned on or off.

EditAppend

Function: Appends selected text or graphics to existing contents of the Windows Clipboard.

Equivalent Command: Edit-Append

Syntax
EditAppend()

Parameters
None

- The Clipboard must already contain data for the **EditAppend** command to work.
- You can append both text and graphics.
- The graphics contained in the Clipboard must have been copied there by WordPerfect.
- Tabular columns and rectangles cannot be appended.
- **EditAppend** will not work if the Clipboard contains tabular columns or rectangles previously copied to it.

EditCopy

Function: Copies selected text or graphics to the Windows Clipboard. The previous contents of the Clipboard are discarded.

Equivalent Command: Edit-Copy

Syntax
EditCopy()

Parameters
None

EditCut

Function: Cuts selected text or graphics and placed it in the Windows Clipboard. The previous contents of the Clipboard are discarded.

Equivalent Command: Edit-Cut

Syntax
EditCut()

Parameters
None

EditPaste

Function: Pastes the contents of the Windows Clipboard at the current insertion point.

Equivalent Command: Edit-Paste

Syntax
EditPaste()

Parameters
None

If text or graphics are selected in the document window, pasting will replace them with the contents of the Clipboard.

EndnoteCreate

Function: Opens a new endnote window.

Equivalent Command: Layout-Endnote-Create

Syntax
EndnoteCreate()

Parameters
None

Actions following this command apply to the Document Endnote window. When you are done, Close to return to the main Edit window.

EndnoteEdit

Function: Edits an existing endnote

Equivalent Command: Endnote Edit dialog options;
Layout-Endnote-Edit

Syntax
EndnoteEdit(
EndnoteNumber:string
)

Parameters
EndnoteNumber: Specifies the number of the endnote to edit. Value is a string; enclose the argument in quotes.

EndnoteEditDlg

Function: Calls the Endnote Edit dialog.

Equivalent Command: Layout-Endnote-Edit

Syntax
EndnoteEditDlg()

Parameters
None

EndnoteNewNumber

Function: Specifies a new starting number for endnotes.

Equivalent Command: Endnote New Number dialog options;
Layout-Endnote-New Number

Syntax
EndnoteNewNumber(
EndnoteNumber:string
)

Parameters EndnoteNumber: Specifies a new starting number for endnotes. Value is a string; enclose the argument in quotes.

EndnoteNewNumberDlg

Function: Calls the Endnote New Number dialog.

Equivalent Command: Layout-Endnote-New Number

Syntax
EndnoteNewNumberDlg()

Parameters
None

EndnoteOptions

Function: Sets options for all subsequent endnotes (or until endnote options are reset).

Equivalent Command: Endnote Options dialog options;
Layout-Endnote-Options

Syntax

```
EndnoteOptions(
        NumberingMethod:{
                Numbers!=0;
                Characters!=1;
                Letters!=2
        };
        Characters:string;
        StyleText:string;
        StyleNote:string;
        LineSpacing:number;
        BetweenSpacing:wpunits;
        MinimumHeight:wpunits
)
```

Parameters

NumberingMethod: Specifies the style for endnote numbers; either numbers, characters, or letters.

Characters: When NumberingStyle: is set to **Characters!**, specifies which character(s) to use in the endnote markers. Maximum length: 5 characters. Value is a string; enclose it in quotes.

StyleText: Specifies how the endnote number appears in the text of the document. (Note: This is an Embedded code.) Enclose the argument in quotes.

StyleNote: Specifies how the endnote number appears in the endnote. (Note: This is an Embedded code.) Enclose the argument in quotes.

LineSpacing: Specifies line spacing within endnotes. Value: any valid spacing setting (whole numbers and decimal numbers), from 0.5 to 160.

BetweenSpacing: Specifies line spacing between endnotes. Value: any valid wpunits.

MinimumHeight: Specifies the minimum amount of an endnote to keep on a page if the note occurs at a page break. Value: any valid wpunits.

EndnoteOptionsDlg

Function: Calls the Endnote Options dialog.

Equivalent Command: Layout-Endnote-Options

Syntax
EndnoteOptionsDlg()

Parameters
None

EndnotePlacement

Function: Forces endnotes that appear up to that point in the document to be printed.

Equivalent Command: Endnote Placement dialog options;
Layout-Endnote-Placement

Syntax
```
EndnotePlacement(
	Restart:{
		No!=0;
		Yes!=1
	}
)
```

Parameters
Restart: Determines whether numbering starts over for subsequent endnotes.

EndnotePlacementDlg

Function: Calls Endnote Placement dialog.

Equivalent Command: Layout-Endnote-Placement

Syntax
EndnotePlacementDlg()

Parameters
None

EquationCaptionEdit

Function: Opens Caption Editor for currently selected or specified equation.

Equivalent Command: Graphics-Equation-Caption

Syntax
EquationCaptionEdit()

Parameters
None

Notes

- If an equation box is currently selected, the caption box for that equation opens.
- If an equation box is not currently selected, the Edit Equation Caption dialog opens, allowing you to insert the number for the equation to edit. The macro then resumes when the caption editor opens.
- Actions following this command apply to the Equation Caption Editor window. When you are done, Close to return to the main Edit window.

EquationCreate

Function: Opens the Equation Editor for a new equation.

Equivalent Command: Graphics-Equation-Create

Syntax
EquationCreate()

Parameters
None

EquationCreateDlg

Function: Calls the Select Editor dialog box.

Equivalent Command: Macro only.

Syntax
EquationCreateDlg()

Parameters
None

EquationEdit

Function: Opens the Equation Editor for currently selected or specified equation.

Equivalent Command: Graphics-Equation-Edit

Syntax
EquationEdit()

Parameters
None

- If an equation box is currently selected, the Equation Editor opens with that equation.
- If an equation box is not currently selected, the Edit Equation dialog opens, allowing you to insert the number for the equation to edit. The macro then resumes when the Equation Editor opens.
- Actions following this command apply to the Equation Editor window. When you are done, Close to return to the main Edit window.

EquationNewNumberDlg

Function: Calls the Equation Box New Number dialog.

Equivalent Command: Graphics-Equation-New Number

Syntax
EquationNewNumberDlg()

Parameters
None

EquationOptionsDlg

Function: Calls the Equation Box Options dialog.

Equivalent Command: Graphics-Equation-Options

Syntax
EquationOptionsDlg()

Parameters
None

EquationPalette

Function: Specifies whether the equation palette in the Equation Editor should be shown or hidden.

Equivalent Command: View-Palette (in Equation Editor)

Syntax

```
EquationPalette(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies display of equation palette.

EquationPositionDlg

Function: Displays the Box Position and Size dialog box for equations. If no equation box is currently selected when this command is encountered, WordPerfect asks the user to enter a box number.

Equivalent Command: Graphics-Equation-Position

Syntax
EquationPositionDlg()

Parameters
None

EquationSettings

Function: Specifies the size and style of the current equation. The Equation Editor should be open before the **EquationSettings** command is used.

Equivalent Command: (In Equation Editor) File-Settings

Syntax

```
EquationSettings(
        PrintAsGraphics:{
                Text!=0;
                Graphics!=1
        };
        FontSize:number;
        HorizontalAlignment:{
                Left!=0;
                Right!=1;
                Center!=2
        };
        VerticalAlignment:{
                Top!=0;
                Center!=1;
                Custom!=2
        }
)
```

Parameters

PrintAsGraphics: Specifies how equation is printed. Choosing **Graphics!** causes graphics to be printed exactly as they appear on screen, but slows printing.

FontSize: Specifies font size to use for equations. If 0, default point size is used. Valid values for specifying another point size are 1 to 1199. Decimal values are allowed.

HorizontalAlignment: Specifies the horizontal alignment of the equation within its box.

VerticalAlignment: Specifies the vertical alignment of the equation within its box.

EquationSettingsDlg

Function: Calls the Equation Settings dialog box.

Equivalent Command: (in Equation Editor) File-Settings

Syntax
EquationSettingsDlg()

Parameters
None

EquationZoom100

Function: Displays equation at 100 percent.

Equivalent Command: (in Equation Editor) View-100%

Syntax
EquationZoom100()

Parameters
None

EquationZoom200

Function: Displays equation at 200 percent.

Equivalent Command: (in Equation Editor) View-200%

Syntax
EquationZomm200()

Parameters
None

EquationZoomFill

Function: Fills equation editor screen with current equation.

Equivalent Command: (in Equation Editor) View-Zoom Fill

Syntax
EquationZoomFill()

Parameters
None

EquationZoomIn

Function: Zooms in for closer magnification of equation

Equivalent Command: (in Equation Editor) View-Zoom In

Syntax
EquationZoomIn()

Parameters
None

EquationZoomOut

Function: Zooms out for broader view of equation.

Equivalent Command: (in Equation Editor) View-Zoom Out

Syntax
EquationZoomOut()

Parameters
None

FigureCaptionEdit

Function: Opens the Caption Editor for the currently selected table box.

Equivalent Command: Graphics-Figure-Caption

Syntax
FigureCaptionEdit()

Parameters
None

FigureCaptionEdit is a non-recordable token. WPWin will record the BoxEditCaption command instead.

FigureCreate

Function: Creates a new figure box and opens the Figure Editor

Equivalent Command: Graphics-Figure-Create

Syntax
FigureCreate()

Parameters
None

FigureCreateDlg

Function: Calls the Select Editor dialog box.

Equivalent Command: Graphics-Figure-Create

Syntax
FigureCreateDlg()

Parameters
None

FigureEdit

Function: Opens the Figure Editor for the currently selected figure box.

Equivalent Command: Graphics-Figure-Edit

Syntax
FigureEdit()

Parameters
None

FigureEdit is a non-recordable token. WPWin will record the BoxEditor command instead.

FigureNewNumberDlg

Function: Calls the Figure Number dialog box.

Equivalent Command: Graphics-Figure-New Number

Syntax FigureNewNumberDlg()

Parameters
None

FigureOptionsDlg

Function: Calls the Figure Options dialog box.

Equivalent Command: Graphics-Figure-Options

Syntax
FigureOptionsDlg()

Parameters
None

FigurePositionDlg

Function: Calls the Box Position and Size dialog.

Equivalent Command: Graphics-Figure-Position

Syntax
FigurePositionDlg()

Parameters
None

FigureRetrieveDlg

Function: Calls the Figure Retrieve dialog box.

Equivalent Command: Graphics-Figure-Retrieve

Syntax
FigureRetrieveDlg

Parameters
None

FileChangeDir

Function: Changes the current default for the indicated directory (makes no change to the Location of Files entrues). Currently, **FileChangeDir** changes two directories, the document and graphics.

Equivalent Command: Macro only.

Syntax

```
FileChangeDir (
        DOSCurrentDir: string;
        WPDirectory:{
                None!=0;
                DocumentsDir!=1;
                GraphicsDir!=2
        }
)
```

Parameters

DOSCurrentDir: Specifies the new directory. The directory must already exist. Note: This value is a string, so enclose it in quotes, or use a string variable.

WPDirectory: Specifies either the documents or graphics directory for changing.

FileCopy

Function: Copies a file.

Equivalent Command: File-Open-[Options]-Copy

Syntax
FileCopy(
    Sourcefile:string;
    Destinationfile:string
)

Parameters
Sourcefile: Specifies the file you want to copy. Enclose the filename (with path, if necessary) in quotes.

Destinationfile: Specifies the name of file you want to create. Enclose the filename (with path, if necessary) in quotes.

FileDelete

Function: Deletes a file.

Equivalent Command: File-Open-[Options]-Delete

Syntax
FileDelete(
Filename:string
)

Parameters
Filename: Specifies the file you want to delete. Enclose the filename (with path, if necessary) in quotes.

FileManager

Function: Starts the WordPerfect File Manager.

Equivalent Command: File-File Manager

Syntax
FileManager()

Parameters
None

FileMove

Function: Moves or renames a file.

Equivalent Command: File-Open-[Options]-Move/Rename

Syntax

```
FileMove(
        SourceFile:string;
        Destinationfile:string
)
```

Parameters

Sourcefile: Specifies the file you want to move/rename. Enclose the filename (with path, if necessary) in quotes.

Destinationfile: Specifies the new name or path of the file. Note: Enclose the filename (with path, if necessary) in quotes.

FileNew

Function: Opens a new, blank document.

Equivalent Command: File-New

Syntax
FileNew()

Parameters
None

WordPerfect supports up to nine open documents at once. The **FileNew** command is ignored if you already have nine open documents.

FileOpen

Function: Opens a previously saved document.

Equivalent Command: File-Open

Syntax

```
FileOpen(
        Filename:string;
        AutoDetect:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Filename: Specifies the name of the file to open. Value is a string; enclose argument in quotes.

AutoDetect: Determines whether WordPerfect automatically determines the file type if the file is not in WP/5.1 format. When set to **No!**, the Convert File Format dialog box is suppressed. When set to **Yes!**, the Convert File Format dialog box is not suppressed should the file be in a format other than WP/5.1.

WordPerfect supports up to nine open documents at once. The **FileOpen** command is ignored if you already have nine open documents.

FileOpenDlg

Function: Calls the File Open dialog.

Equivalent Command: File-Open.

Syntax
FileOpenDlg()

Parameters
None

FileRetrieve

Function: Retrieves the contents of another file into the current document. The file is inserted at the insertion point.

Equivalent Command: File-Retrieve

Syntax

```
FileRetrieve(
        Filename:string;
        AutoDetect:{
                No!=0;
                Yes!=1
        };
        InsertIntoDoc:{
                Prompt!=0;
                Insert!=1
        }
)
```

Parameters

Filename: Specifies the name of the file to retrieve. Value is a string; enclose argument in quotes.

AutoDetect: Determines whether WordPerfect automatically determines the file type if the file is not in WP/5.1 format. When set to **No!**, the Convert File Format dialog box is suppressed. When set to **Yes!**, the Convert File Format dialog box is not suppressed should the file be in a format other than WP/5.1.

InsertIntoDoc: If the current document has been used (includes typing or formatting), WordPerfect will insert the retrieved document at the current insertion point. When InsertIntoDoc is set to **Prompt!**, WordPerfect displays a dialog box to ask permission to retrieve the document. When InsertIntoDoc is set to **Insert!**, the dialog box is suppressed.

FileRetrieveDlg

Function: Calls the File Retrieve dialog.

Equivalent Command: File-Retrieve

Syntax
FileRetrieveDlg()

Parameters
None

FileSave

Function: Saves changes to the current document.

Equivalent Command:

Syntax

```
FileSave(
        FileName:string;
        ExportType:{
                WordPerfect42!=1;
                WordPerfect50!=2;
                WordPerfect51!=3;
                WordStar33!=30;
                WordStar331!=31;
                WordStar34!=32;
                WordStar40!=33;
                WordStar50!=34;
                WordStar55!=35;
                WordStar60!=36;
                MicrosoftWord50!=63;
                MicrosoftWord55!=64;
                WordForWindows10!=70;
                WordForWindows11!=71;
                IBMDCARevisableFormText!=80;
                IBMDCAFinalFormText!=81;
                DisplayWrite40!=82;
                DisplayWrite42!=83;
                DisplayWrite50!=84;
                MultiMate33!=100;
                MultiMateAdvantage36!=101;
                MultiMateAdvantageII37!=102;
                MultiMate40!=103;
                OfficeWriter60!=120;
                OfficeWriter61!=121;
                OfficeWriter611!=122;
                OfficeWriter62!=123;
                WordRichTextFormat!=150;
                XyWriteIIIPlus355!=190;
                XyWriteIIIPlus356!=191;
                AmiPro12Windows!=311;
                AmiPro12aWindows!=312;
                AmiPro12bWindows!=313;
                ANSIDelimitedTextWindows!=487;
                StrippedANSIWindows!=488
        };
        Overwrite:{
                No!=0;
                Yes!=1;
                Prompt!=2
    }
)
```

Parameters
FileName: Names the file. WordPerfect always displays the File Save dialog box to confirm the name. The FileName argument is a string; enclose argument in quotes.

ExportType: Specifies the file format.

Overwrite: Speciifies if WPWin prompts or automatically overwrites an existing file.

FileSaveAsDlg

Function: Calls the File Save As dialog.

Equivalent Command: File-Save As

Syntax
FileSaveAsDlg()

Parameters
None

Font

Function: Changes the current font at the insertion point. Text entered from that point will be displayed and printed in the new font specified.

Equivalent Command: Font-Font-{**any font**}

Syntax

```
Font(
        Name:string;
        Size:number;
        ForceInsert:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Name: Specifies the unique name of the font. Value is a string; enclose argument in quotes.

Size: Specifies the size of the font, from 1 to 1199 points.

ForceInsert: Specifies whether you want the font code to be inserted into the document even if you are specifying the same font already being used.

- Name should correspond to the actual name of a font for the currently selected printer. If you're not sure of the correct name, check first.
- Not all fonts are available in all sizes. Verify first that a font is available in a certain size before specifying it in the **Font** command.

FontColor

Function: Specifies a color for text.

Equivalent Command: Font Color dialog option;
Font-Color

Syntax
FontColor(
AmountRed:number;
AmountGreen:number;
AmountBlue:number;
)

Parameters
AmountRed, AmountGreen, AmountBlue: Sets amount of intensity for specified colors. Value: 0 to 255 (whole numbers only). Use combinations of colors for up to 1.6 million combinations. A setting of 0 is no intensity; 255 is full intensity.

FontColorDlg

Function: Calls the Color dialog.

Equivalent Command: Font-Color

Syntax
FontColorDlg()

Parameters
None

FontDlg

Function: Calls the Font dialog.

Equivalent Command: Font-Font

Syntax
FontDlg()

Parameters
None

FontAttribute

Function: Forces the specified font attribute on or off. See Notes for available attribute types.

Equivalent Command: See each attribute tyoe below;
also available through Font-Font dialog

Syntax

```
FontAttribute(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies the state of the font attribute.

Available FontAttribute types:

- FontBold (Font-Bold)
- FontDoubleUnderline (Font-Double Underline)
- FontExtraLarge (Font-Size-Extra Large)
- FontFine (Font-Size-Fine)
- FontItalic (Font-Italic)
- FontLarge (Font-Size-Large)
- FontOutline (Font-Font-Outline)
- FontRedline (Font-Redline)
- FontShadow (Font-Font-Shadow)
- FontSmallCaps (Font-Font-Small Caps)
- FontStrikeout (Font-Strikeout)
- FontSubscript (Font-Subscript)
- FontSuperscript (Font-Superscript)
- FontUnderline (Font-Underline)
- FontVeryLarge (Font-Size-Very Large)

FontNormal

Function: Turns off all font attributes.

Equivalent Command: Font-Normal

Syntax
FontNormal()

Parameters
None

FooterDlg

Function: Calls the Footers dialog.

Equivalent Command: Layout-Page-Footers

Syntax
FooterDlg()

Parameters
None

FootnoteCreate

Function: Opens a footnote window.

Equivalent Command: Layout-Footnote-Create

Syntax
FootnoteCreate()

Parameters
None

Actions following this command apply to the Footnote window. When you are done, Close to return to the main Edit window.

FootnoteEdit

Function: Opens the footnote windows for the specified footnote.

Equivalent Command: Layout-Footnote-Edit

Syntax
FootnoteEdit(
FootnoteNumber:string
)

Parameters
FootnoteNumber: Specifies the number of the footnote you want to edit. Value is a string; enclose argument in quotes.

Actions following this command apply to the Footnote window. When you are done, Close to return to the main Edit window.

FootnoteEditDlg

Function: Calls the Edit Footnote dialog.

Equivalent Command: Layout-Footnote-Edit

Syntax
FootnoteEditDlg()

Parameters
None

FootnoteNewNumber

Function: Specifies a new starting number for footnotes.

Equivalent Command: Layout-Footnote-New Number

Syntax
FootnoteNewNumber(
FootnoteNumber:string
)

Parameters
FootnoteNumber: Specifies the number of the footnote you want to edit. Value is a string; enclose the argument in quotes.

FootnoteNewNumberDlg

Function: Calls Footnote Number dialog.

Equivalent Command: Layout-Footnote-New Number

Syntax
FootnoteNewNumberDlg()

Parameters
None

FootnoteOptions

Function: Sets all options for all subsequent footnotes.

Equivalent Command: Layout-Footnote-Options

Syntax

```
FootnoteOptions(
        NumberingMethod:{
                Numbers!=0;
                Characters!=1;
                Letters!=2
        };
        Characters:string;
        StyleText:string;
        StyleNote:string;
        LineSpacing:number;
        BetweenSpacing:wpunits;
        MinimumHeight:wpunits;
        Restart:{
                Off!=0;
                On!=1
        };
        PrintContinued:{
                No!=0;
                Yes!=1
        };
        FootnotePosition:{
                AfterText!=0;
                BottomOfPage!=1
        };
        Separator:{
                NoLine!=0;
                ShortLine!=1;
                MarginToMargin!=2
        }
)
```

Parameters

NumberingMethod: Specifies markers used to identify footnotes: numbers, characters, or letters.

Characters: When NumberingMethod is set to **Characters!,** specifies one or more characters to use to identify a footnote. The same character(s) are then used for all footnotes. Value is a string; enclose parameter in quotes.

StyleText: Specifies how the footnote number appears in the text of the document. (Note: This is an Embedded code.) Enclose the argument in quotes.

StyleNote: Specifies how the footnote number appears in the endnote. (Note: This is an Embedded code.) Enclose the argument in quotes.

LineSpacing: Specifies line spacing within footnotes. Value: any valid spacing setting (whole numbers and decimal numbers), from 0.5 to 160.

BetweenSpacing: Specifies line spacing between body of document and footnotes. Value: any valid WordPerfect measurement unit.

MinimumHeight: Specifies the minimum number of a footnote to keep together on a page. Value: any valid WordPerfect measurement unit.

Restart: Specifies whether footnote numbering restarts at 1 (or A when using letters) for each new page.

PrintContinued: Specifies "continued" message for footnotes that span more than one page.

FootnotePosition: Specifies location of footnotes when printed, either after all the text in the document, or at the bottom of each page.

Separator: Specifies the type of line used to separate the body of the document and footnotes.

FootnoteOptionsDlg

Function: Calls the Footnote Options dialog.

Equivalent Command: Layout-Footnote-Options

Syntax
FootnoteOptionsDlg()

Parameters
None

ForcePageEven

Function: Forces the current page (the one with the insertion point) to be an even-numbered page when printed. If the page would otherwise be an odd-numbered page, that page is left blank, and the text appears on the following page.

Equivalent Command: Layout-Page-Numbering-(Even)

Syntax
ForcePageEven()

Parameters
None

ForcePageOdd

Function: Forces the current page (the one with the insertion point) to be an odd-numbered page when printed. If the page would otherwise be an even-numbered page, that page is left blank, and the text appears on the following page.

Equivalent Command: Layout-Page-Numbering-(Odd)

Syntax
ForcePageOdd()

Parameters
None

Generate

Function: Generates the document so that tables, lists, and indexes are updated.

Equivalent Command: Tools-Generate

Syntax
Generate()

Parameters
None

GenerateDlg

Function: Calls the Generate dialog box.

Equivalent Command: Tools-Generate

Syntax
GenerateDlg()

Parameters
None

GetWPData

Function: Queries WordPerfect on the status of a variety of system-level functions and values, including the name of the current document, the location of the insertion point, the font attributes currently active, and so forth.

Equivalent Command: Macro only.

Syntax

```
GetWPData(
        MacroVariable:variable;
        SystemVariable:{
                Cell!=2;
                Column!=3;
                Endnote!=5;
                Equation!=6;
                Figure!=7;
                Footnote!=8;
                LeftCode!=9;
                Line!=10;
                Name!=12;
                Page!=14;
                Path!=15;
                Pos!=16;
                RightCode!=18;
                TableBox!=19;
                TextBox!=20;
                UserBox!=21;
                Row!=22;
                Rowstate!=27;
                Network!=30;
                Language!=32;
                FontNormal!=64;
                FontExtraLarge!=65;
                FontVeryLarge!=66;
                FontLarge!=67;
                FontSmall!=68;
                FontFine!=69;
                FontSuperscript!=70;
                FontSubscript!=71;
                FontOutline!=72;
                FontItalics!=73;
                FontShadow!=74;
                FontRedline!=75;
                FontDoubleUnderline!=76;
                FontBold!=77;
                FontStrikeout!=78;
                FontUnderline!=79;
                FontSmallCaps!=80;
                CellNormal!=81;
                CellExtraLarge!=82;
                CellVeryLarge!=83;
                CellLarge!=84;
                CellSmall!=85;
                CellFine!=86;
```

```
                CellSuperscript!=87;
                CellSubscript!=88;
                CellOutline!=89;
                CellItalics!=90;
                CellShadow!=91;
                CellRedline!=92;
                CellDoubleUnderline!=93;
                CellBold!=94;
                CellStrikeout!=95;
                CellUnderline!=96;
                CellSmallCaps!=97;
                DocumentModified!=98;
                DocumentNeedsGenerating!=99;
                DocumentBlank!=100;
                BetweenTableCodes!=101;
                BetweenMathCodes!=102;
                BetweenOutlineCodes!=103;
                CellJustification!=104;
                JustifyCellSpecific!=105;
                AttributeCellSpecific!=106;
                CellBottomAligned!=107;
                CellCenterAligned!=108;
                CellIgnoreWhileCalculating!=109;
                CellContentIsFormula!=110;
                CellLocked!=111;
                LeftChar!=112
                RightChar!=113
                MacroPath!=114
                SharedPath!=115
                BetweenColumnCodes!=116
                CurrentDocument!=128;
                AtMainEditScreen!=129;
                MacroDefActive!=130;
                MacroExecuteActive!=131;
                MergeActive!=132;
                SelectModeActive!=133;
                TypeoverMode!=134;
                RevealCodesActive!=135;
                MajorVersion!=136;
                MinorVersion!=137;
                InterimRelease!=138;
                AutoCodePlacement!=139;
                ConfirmationOnCodeDelete!=140;
                ReadOnlyDoc!=141;
                DocSummaryPromptOnExit!=142
                SelectedText!=143;
                SelectedTextWithHRt!=144
        }
)
```


Parameters

MacroVariable: Specifies the macro variable to hold the return value of the desired SystemVariable.

SystemVariable: Specifies the name of the system variable you want to check.

- Some system variables return strings, and some return numeric values. You must take the variable type into consideration if it is used in other macro functions. For example, you cannot compare a number with a system variable that returns a string. Likewise, you can't Type a numeric system variable without first converting it using the NumStr programming command.
- WordPerfect occasionally adds system variables to interim releases of WordPerfect. You should check the Software Change Notice that comes with the software for an updated list.
- A complete run-down of the system variables, what they do, and how to use them, can be found in **WordPerfect for Windows Power Tools**.

GraphicLine

Function: Creates a horizontal or vertical graphic line, of a specified width, length, position, and shade.

Equivalent Command: Graphic-Line-Create Horizontal or
Graphic-Line-Create Vertical

Syntax

```
GraphicLine(
        Operation:{
                HCreate!=0;
                VCreate!=1;
                Edit!=3
        };
        LineLength:wpunits;
        Thickness:wpunits;
        Shading:number;
        VerticalType:{
                Baseline!=0;
                FullPage!=1;
                Top!=2;
                Center!=3;
                Bottom!=4;
                Specify!=5
        };
        VerticalPosition:wpunits;
        HorizontalType:{
                Left!=0;
                Right!=1;
                Center!=2;
                Full!=3;
                Columns!=4;
                Specify!=5
        };
        HorizontalPosition:wpunits;
        Column:number
)
```

Parameters

Operation: Specifies the type of line to create, either horizontal or vertica, or specifies that you want to edit the currently selected graphic line.

LineLength: Specifies the length of the line. Value: any valid wpunits measurement.

Thickness: Specifies the thickness of the line. Value: any valid wpunits measurement.

Shading: Specifies the shading of the line, in percent. Value: from 0 (white) to 100 (black).

VerticalType: Specifies vertical alignment.

VerticalPosition: When VerticalType is set to **Specify!**, indicates the actual vertical position of the line. Value: any valid wpunits measurement.

HorizontalType: Specifies horizontal alignment.

HorizontalPosition: When HorizontalType is set to **Specify!**, indicates the actual horizontal position of the line. Value: any valid wpunits measurement.

Column: Specifies the column for a column line. The line will appear to the left of that column.

GraphicLineSelect

Function: Selects the next graphic line of the specified type. If no graphic line can be found forward of the insertion point, WordPerfect selects the first graphic line it finds behind the insertion point.

Equivalent Command: Macro only.

Syntax

```
GraphicLineSelect(
        Type:{
                Horizontal!=0;
                Vertical!=1
        }
)
```

Parameters

Type: Specifies the type of line to select.

GraphicsShow

Function: Shows or hides graphics in documents. With GraphicsShow off, documents are displayed more quickly.

Equivalent Command: View-Graphics

Syntax

```
GraphicsShow(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies whether graphics are shown or hidden.

HardHyphen

Function: Inserts a hard hyphen at the insertion point. The code is entered as [-].

Equivalent Command: Hyphen (**-**).

Syntax
HardHyphen()

Parameters
None

HardPageBreak

Function: Inserts a hard page break at the insertion point.

Equivalent Command: **[Ctrl+Enter]**

Syntax
HardPageBreak()

Parameters
None

HardPageBreakInsert

Function: Used with the outline feature, inserts a new page without inserting a paragraph number in the next page.

Equivalent Command: Macro

Syntax
HardPageBreakInsert()

Parameters
None

HardReturn

Function: Inserts a hard return in the document.

Equivalent Command: **[Enter]**

Syntax
HardReturn()

Parameters None

HardReturnInsert

Function: Used with the outline feature, inserts a hard return at the end of the line without inserting a paragraph number.

Equivalent Command: **[Shift+Enter]**

Syntax
HardReturnInsert()

Parameters
None

HardSpace

Function: Inserts a hard space in the document.

Equivalent Command: **[Ctrl+Space]**

Syntax
HardSpace()

Parameters
None

HardTabCenter

Function: Inserts a hard centered tab in the document.

Equivalent Command: Layout-Line-Special Codes-(Center)

Syntax
HardTabCenter()

Parameters
None

HardTabCenterDot

Function: Inserts a hard centered tab (with dot leader) in the document.

Equivalent Command: Layout-Line-Special Codes-(Center)

Syntax
HardTabCenterDot()

Parameters
None

HardTabDecimal

Function: Inserts a hard decimal tab in the document.

Equivalent Command: Layout-Line-Special Codes-(Decimal)

Syntax
HardTabDecimal()

Parameters
None

HardTabDecimalDot

Function: Inserts a hard decimal tab (with dot leader) in the document.

Equivalent Command: Layout-Line-Special Codes-(Decimal)

Syntax
HardTabDecimalDot()

Parameters
None

HardTabDecimalInsert

Function: Inserts a decimal tab at the insertion point even if there are no more tab stops specified for the line

Equivalent Command: Macro only.

Syntax
HardTabDecimalInsert()

Parameters
None

HardTabLeft

Function: Inserts a hard left tab in the document.

Equivalent Command: Layout-Line-Special Codes-(Left)

Syntax
HardTabLeft()

Parameters
None

HardTabLeftDot

Function: Inserts a hard left tab (with dot leader) in the document.

Equivalent Command: Layout-Line-Special Codes-(Left)

Syntax
HardTabLeftDot()

Parameters
None

HardTabRight

Function: Inserts a hard right tab in the document.

Equivalent Command: Layout-Line-Special Codes-(Right)

Syntax
HardTabRight()

Parameters
None

HardTabRightDot

Function: Inserts a hard right tab (with dot leader) in the document.

Equivalent Command: Layout-Line-Special Codes-(Right)

Syntax
HardTabRightDot()

Parameters
None

HeaderDlg

Function: Calls the Headers dialog.

Equivalent Command: Layout-Page-Headers

Syntax
HeaderDlg()

Parameters
None

HeaderFooter

Function: Creates or edits headers or footers.

Equivalent Command: Layout-Page-Headers or
Layout-Page-Footers

Syntax

```
HeaderFooter(
        Operation:{
                Create!=0;
                Discontinue!=1;
                Edit!=2
        };
        Item:{
                HeaderA!=0;
                HeaderB!=1;
                FooterA!=2;
                FooterB!=3
        }
)
```

Parameters

Operation: Specifies the operation for the header or footer: create, discontinue, or edit.

Item: Specifies the header or footer.

HeaderFooterPlacement

Function: Specifies how the currently edited header or footer is to be printed: on every page, on odd pages only, or on even pages only.

Equivalent Command: (in Header/Footer window) [Placement]

Syntax

```
HeaderFooterPlacement(
        State:{
                EveryPage!=0;
                EvenPages!=1;
                OddPages!=2
        }
)
```

Parameters

State: Specifies printing frequency for the header or footer.

HeaderFooterPlacementDlg

Function: Calls the Placement dialog box.

Equivalent Command: (in Header/Footer window) [Placement]

Syntax
HeaderFooterPlacementDlg()

Parameters
None

HelpContextSensitive

Function: Invokes Help

Equivalent Command: **[F1]**

Syntax
HelpContextSensitive()

Parameters
None

HelpGlossary

Function: Calls the Glossary function of the Help facility.

Equivalent Command: Help-Glossary

Syntax
HelpGlossary()

Parameters
None

HelpHowDoI

Function: Calls the How Do I function of the Help facility.

Equivalent Command: Help-How Do I?

Syntax
HelpHowDoI()

Parameters
None

HelpIndex

Function: Calls the Index function of the Help facility.

Equivalent Command: Help-Index

Syntax
HelpIndex()

Parameters
None

HelpKeyboard

Function: Calls the Keyboard function of the Help facility.

Equivalent Command: Help-Keyboard

Syntax
HelpKeyboard()

Parameters
None

HelpUsingMacros 

Function: Calls the Using Macros function of the Help facility.

Equivalent Command: Help-Using Macros

Syntax
HelpUsingMacros()

Parameters
None

HelpUsingWPHelp

Function: Calls the Using Help function of the Help facility.

Equivalent Command: Help-Using Help

Syntax
HelpUsingWPHelp()

Parameters
None

HelpWhatIs

Function: Calls the What Is function of the Help facility.

Equivalent Command: Help-What Is or **[Shift+F1]**

Syntax
HelpWhatIs()

Parameters
None

HorizLineCreateDlg

Function: Calls the Create Horizontal Line dialog box.

Equivalent Command: Graphics-Line-Horizontal

Syntax
HorizLineCreateDlg()

Parameters
None

HorizLineEditDlg

Function: Calls the Edit Horizontal Line dialog box.

Equivalent Command: Graphics-Line-Edit Horizontal

Syntax
HorizLineEditDlg()

Parameters
None

HorizontalScrollBarShow

Function: Displays or hides the horizontal scroll bar.

Equivalent Command: View-Horizontal Scroll

Syntax:
```
HorizontalScrollBarShow (
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters:
State: Specifies if the horizontal scroll bar is displayed or hidden.

Hyphen

Function: Inserts a hyphen.

Equivalent Command: **[Ctrl+-]** (**[Ctrl]** and hyphen)

Syntax
Hyphen()

Parameters
None

HyphenIgnoreWord

Function: Inserts a hyphen ignore word (hyphen cancel) code.

Equivalent Command: Layout-Line-Special Codes-
(Hyphenation Ignore Word)

Syntax
HyphenIgnoreWord()

Parameters
None

HyphenSoft

Function: Inserts a soft hyphen

Equivalent Command: **[Ctrl+Shift+-]** (Ctrl and Shift and hyphen)

Syntax
HyphenSoft()

Parameters
None

HyphenSoftReturn

Function: Inserts a hyphen soft return code.

Equivalent Command: Layout-Line-Special Codes-
(Hyphenation Soft Return)

Syntax
HyphenSoftReturn()

Parameters
None

IndexDefine

Function: Defines a new index. A code is inserted at the insertion point; the index when generated will appear at the code.

Equivalent Command: Tools-Define-Index

Syntax IndexDefine(
NumberingStyle:{
NoNumbering!=0;
NumFollowsEntry!=1;
FollowsInParens!=2;
FlushRight!=3;
FlushLeaders!=4
};
ConcordanceFile:string
)

Parameters
NumberingStyle: Specifies the numbering style for the index: no numbering, number follows index entry, number follows entry in parentheses, number flush right, number flush right with dot leader.

ConcordanceFile: Specifies an optional concordance file. Enclose filename in quotes (include path if necessary).

IndexDefineDlg

Function: Calls the Define Index dialog box.

Equivalent Command: Tools-Define-Index

Syntax
IndexDefineDlg()

Parameters
None

IndexMark

Function: Marks an item in the document for indexing.

Equivalent Command: Tools-Mark-Index

Syntax

```
IndexMark(
        HeadingText:string;
        SubheadingText:string
)
```

Parameters

HeadingText: Specifies the heading for the index entry. Enclose string in quotes.

SubheadingText: Specifies the subheading for the index entry. Enclose string in quotes.

IndexMarkDlg

Function: Calls the Mark Index dialog box.

Equivalent Command: Tools-Mark-Index

Syntax
IndexMarkDlg()

Parameters
None

InsertSpecialCodesDlg

Function: Calls the Insert Special Codes dialog.

Equivalent Command: Layout-Line-Special Codes

Syntax InsertSpecialCodesDlg()

Parameters
None

InsertTypeover

Function: Turns Insert mode on or off.

Equivalent Command: **[Ins]**

Syntax

```
InsertTypeover(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Turns typeover mode on or off.

JustifyFormat

Function: Formats subsequent text centered, left, right, or full.

Equivalent Command: See below.

Syntax

```
JustifyFormat(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies the state of the font attribute.

Available JustifyFormat types

- JustifyCenter (Layout-Justification-Center)
- JustifyFull (Layout-Justification-Full)
- JustifyLeft (Layout-Justification-Left)
- JustifyRight (Layout-Justification-Right)

KerningSpacing

Function: Sets manual kerning spacing for the next character.

Equivalent Command: Layout-Typesetting-[Manual Kerning]

Syntax
KerningSpacing(
KerningSpacingValue:wpunits
)

Parameters
KerningSpacingValue: Specifies the kerning spacing. Value: any valid wpunits measurement.

KeyboardSelect

Function: Selects the named keyboard and makes it active.

Equivalent Command: File-Preferences-Keyboard

Syntax
KeyboardSelect(
KeyboardFile:string
)

Parameters
KeyboardFile: Name of the keyboard file. Enclose filename in quotes (include path if necessary).

KeyboardSelectDlg

Function: Calls the Select Keyboard dialog box.

Equivalent Command: File-Preferences-Keyboard

Syntax
KeyboardSelectDlg()

Parameters
None

Language

Function: Specifies the current language module to use.

Equivalent Command: Tools-Language

Syntax
Language(
Language:string
)

Parameters
Language: Specifies the language to use (always two characters). Enclose string in quotes

LanguageDlg

Function: Calls the Language dialog box.

Equivalent Command: Tools-Language

Syntax
LanguageDlg()

Parameters
None

LineCenter

Function: Centers all text from the insertion point to the end of the line.

Equivalent Command: Layout-Line-Center

Syntax
LineCenter()

Parameters
None

LineCenterEnd

Function: Ends line centering without inserting a hard return.

Equivalent Command: Layout-Line-Special Codes-
(End Centering/Alignment)

Syntax
LineCenterEnd()

Parameters
None

LineDraw

Function: Draws a line of specified type, length, and direction.

Equivalent Command: Tools-Line Draw

Syntax

```
LineDraw(
        Item:{
                ItemLineSingle!=0;
                ItemLineDouble!=1;
                ItemBlockHatchedLeft!=2;
                ItemBlockHatched!=3;
                ItemBlockHatchedRight!=4;
                ItemBlockSolid!=5;
                ItemBlockSolidBottom!=6;
                ItemBlockSolidLeft!=7;
                ItemBlockSolidRight!=8;
                ItemBlockSolidTop!=9;
                ItemCharacter!=10;
                ItemMove!=11;
                ItemErase!=12;
                ItemEnd!=13
        };
        Direction:{
                PosCharPrev!=0;
                PosCharNext!=1;
                PosLineUp!=2;
                PosLineDown!=3;
                PosWordPrev!=4;
                PosWordNext!=5;
                PosParagraphNext!=6;
                PosParagraphPrev!=7;
                PosLineBegin!=8;
                PosLineEnd!=9
        };
        Character:string;
        RepeatCount:number
)
```

Parameters

Item: Specifies the character to use.

Direction: Specifies direction of line drawing.

Character: When Item is set to **ItemCharacter!**, specifies the WordPerfect character to use. Enclose string in quotes.

RepeatCount: Specifies the number of times the character is repeated to build the line. Use when Direction: is set to move a single line or character at a time. RepeatCount: defaults to 1.

LineDrawDlg

Function: Calls the Line Draw dialog box.

Equivalent Command: Tools-Line Draw

Syntax
LineDrawDlg()

Parameters
None

LineFlushRight

Function: Sets line flush to the right margin.

Equivalent Command: Layout-Line-Flush Right

Syntax
LineFlushRight()

Parameters
None

LineHeight

Function: Sets manual line height (overrides automatic line height control for font changes), or returns to automatic line height.

Equivalent Command: Layout-Line-Height

Syntax

```
LineHeight(
        Type:{
                Auto!=0;
                Fixed!=1
        };
        Height:wpunits
)
```

Parameters

Type: Species the new setting for line height.

Height: When Type: is set to **Fixed!**, specifies manual line height adjustment. Value: any valid wpunits measurement.

LineHeightDlg

Function: Calls the Line Height dialog box.

Equivalent Command: Layout-Line-Height

Syntax
LineHeightDlg()

Parameters
None

LineHyphenation

Function: Specifies the hyphenation zone.

Equivalent Command: Layout-Line-Hyphenation

Syntax

```
LineHyphenation(
        State:{
                Off!=0;
                On!=1
        };
        PercentLeft:number;
        PercentRight:number
        }
)
```

Parameters

State: Specifies whether line hyphenation is turned on or off.

PercentLeft: Specifies left hyphenation zone (whole number only, from 0 to 100).

PercentRight: Specifies the right hyphenation zone (whole number only, from 0 to 100).

LineHyphenationDlg

Function: Calls the Line Hyphenation dialog box.

Equivalent Command: Layout-Line-Hyphenation

Syntax
LineHyphenationDlg()

Parameters
None

LineNumbering

Function: Controls line numbering.

Equivalent Command: Layout-Line-Numbering

Syntax

```
LineNumbering(
        State:{
                Off!=0;
                RestartEachPage!=1;
                Continuous!=2
        };
        Start:number;
        Increment;number;
        NumberingPosition:wpunits;
        BlankLines:{
                Skip!=0;
                Count!=1
        }
)
```

Parameters

State: Specifies if line numbering is off, restarts for each page, or continues for subsequent pages.

Start: Specifies a starting line number, from 0 to 65535.

Increment: Specifies numbering for every x lines, from 0 to 30.

NumberingPosition: Specifies numbering position. Value: any valid wpunits measurement.

BlankLines: Specifies how blank lines (containing only a hard return) are to be treated.

LineNumberingDlg

Function: Calls the Line Numbering dialog box.

Equivalent Command: Layout-Line-Numbering

Syntax
LineNumberingDlg()

Parameters
None

LineSpacing

Function: Specifies line spacing, in whole or fractional lines.

Equivalent Command: Layout-Line-Spacing

Syntax
LineSpacing(
Spacing:number
)

Parameters
Spacing: Specifies the line spacing. Valid values are 0.5 to 160.

LineSpacingDlg

Function: Calls the Line Spacing dialog box.

Equivalent Command: Layout-Line-Spacing

Syntax
LineSpacingDlg()

Parameters
None

ListDefine

Function: Defines a new list. A code is inserted at the insertion point; the list when generated will appear at the code.

Equivalent Command: Tools-Define-List

Syntax

```
LineDefine(
        ListType:{
                UserList1!=0;
                UserList2!=1;
                UserList3!=2;
                UserList4!=3;
                UserList5!=4;
                FigureCaptions!=5;
                TableCaptions!=6;
                TextBoxCaptions!=7;
                UserBoxCaptions!=8;
                EquationCaptions!=9
        };
        NumberingStyle:{
                NoNumbering!=0;
                NumFollowsEntry!=1;
                FollowsInParens!=2;
                FlushRight!=3;
                FlushLeaders!=4
        }
)
```

Parameters

ListType: Specifies the number of the list to create.

NumberingStyle: Specifies the numbering style for the list: no numbering, number follows index entry, number follows entry in parentheses, number flush right, number flush right with dot leader.

ListDefineDlg

Function: Calls the Define List dialog box.

Equivalent Command: Tools-Define-List

Syntax
ListDefineDlg()

Parameters
None

ListMark

Function: Marks current selection for the specified list.

Equivalent Command: Tools-Mark Text-List

Syntax

```
ListMark(
        ListType:{
                UserList1!=0;
                UserList2!=1;
                UserList3!=2;
                UserList4!=3;
                UserList5!=4;
                FigureCaptions!=5;
                TableCaptions!=6;
                TextBoxCaptions!=7;
                UserBoxCaptions!=8;
                EquationCaptions!=9
        }
)
```

Parameters

ListType: Specifies which list the current selected will be marked.

ListMarkDlg

Function: Calls the Mark List dialog box.

Equivalent Command: Tools-Mark Text-List

Syntax
ListMarkDlg()

Parameters
None

MacroAssignDlg

Function: Calls the Assign to Macro to Menu dialog box.

Equivalent Command: Macro-Assign to Menu

Syntax
MacroAssignDlg()

Parameters
None

MacroMenuAppend

Function: Appends up to nine existing macros to the Macro menu.

Equivalent Command: Macro-Assign to Menu-[Add]

Syntax

```
MacroMenuAppend(
        MenuText1:string;
        MacroMenu1:string;      ...
        MacroText9:string;
        MacroMenu9:string
)
```

Parameters

MenuText**x**: Specifies the text for the macro in the Macro menu (1 through 9). Enclose string in quotes.

MacroMenu**x**: Specifies the filename of the macro to use. Enclose filename in quotes (include path if necessary).

MacroMenuDelete

Function: Deletes a macro from the Macro menu.

Equivalent Command: Macro-Assign to Menu-[Delete]

Syntax
MacroMenuDelete(
MacroNumber:number
)

Parameters
MacroNumber: The number of the macro item to delete. Valid values: 1 through 9.

MacroStatusPrompt

Function: Displays or hides a one-line message in the status bar.

Equivalent Command: Macro only.

Syntax

```
MacroStatusPrompt(
        State:{
                Off!=0;
                On!=1
        };
        Prompt:string
)
```

Parameters

State: Turns the message display on or off.

Prompt: Specifies the message for the status line. Enclose the string in quotes. When State: is **Off!**, the prompt message is ignored.

MarginRelease

Function: When the insertion point is outside a table or outline, moves it past the left margin or to the previous tab stop. When the insertion point is inside a table, moves it to the previous cell. The insertion point is inside an outline, moves it to the previous level.

Equivalent Command: **[Shift+Tab]**

Syntax
MarginRelease()

Parameters
None

MarginReleaseInsert

Function: Inserts a margin release code. See MarginRelease for the different effect of the margin release depending on the location of the insertion point.

Equivalent Command: **[Ctrl+Shift+Tab]**

Syntax
MarginReleaseInsert()

Parameters
None

MasterDocCondense

Function: Condenses a previously expanded document.

Equivalent Command: Tools-Master Document-Condense Master

Syntax

```
MasterDocCondense(
        SaveSubdocuments:(
                No!=0;
                Yes!=1
        };
        ConfirmOverwrite:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

SaveSubdocuments: Specifies whether changes to any subdocument text should be saved.

ConfirmOverwrite: Specifies whether you will be prompted to save the changes to individual subdocuments.

Condensing a master document can take time, particularly if changes to the sub-documents are saved. Macro execution is paused until the document is completely condensed.

MasterDocCondenseDlg

Function: Calls the Condense Master Document dialog box.

Equivalent Command: Tools-Master Document-Condense Master

Syntax
MasterDocCondenseDlg()

Parameters
None

MasterDocExpand

Function: Expands a document with subdocument codes to its full form.

Equivalent Command: Tools-Master Document-Expand Document

Syntax
MasterDocExpand()

Parameters
None

MasterDocSubdocDlg

Function: Calls the Include Subdocument dialog box.

Equivalent Command: Tools-Master Document-Subdocument

Syntax MasterDocSubdocDlg()

Parameters
None

MasterDocSubdocInsert

Function: Inserts a subdocument code.

Equivalent Command: Tools-Master Document-Subdocument

Syntax
MasterDocSubdocInsert(
SubdocumentName:string
)

Parameters
SubdocumentName: Specifies the name of the subdocument to use. Enclose filename in quotes (include path if necessary).

MergeCodesDlg

Function: Calls the Insert Merge Codes dialog.

Equivalent Command: Tools-Merge-Merge Codes

Syntax
MergeCodesDlg()

Parameters
None

MergeConvert

Function: Converts WordPerfect 4.2 or 5.0 merge codes to WordPerfect 5.1 format.

Equivalent Command: Tools-Merge-Convert

Syntax
MergeConvert()

Parameters
None

MergeConvertMsgDlg

Function: Calls the Convert Old Merge Codes dialog box. This command performs the same function as MergeConvert, but offers the user the chance to decide whether to convert the merge codes.

Equivalent Command: Tools-Merge-Convert

Syntax
MergeConvertMsgDlg()

Parameters
None

MergeDOSText

Function: Specifies record and field delimiters when using DOS text files as secondary merge documents.

Equivalent Command: Tools-Merge-Merge-
(Ascii Delimited Text)

Syntax
MergeDOSText(
FieldBegin:string;
FieldEnd:string; RecordBegin:string;
RecordEnd:string
)

Parameters
FieldBegin: Specifies the character to delimit the beginning of fields.

FieldEnd: Specifies the character to delimit the end of fields.

RecordBegin: Specifies the character to delimit the beginning records.

RecordEnd: Specifies the characters to delimit the end of records.

- All parameters use string values. Be sure to enclose the strings in quotes.
- You can specify more than one character as a field or record delimiter.
- If you want to use a formatting code, such as a hard return or tab, you need to insert an "embedded" code as the string value.

MergeEndField

Function: Inserts an {END MERGE} code.

Equivalent Command: Tools-Merge-End Field

Syntax
MergeEndField()

Parameters
None

MergeEndRecord

Function: Inserts an {END RECORD} code.

Equivalent Command: Tools-Merge-End Record

Syntax
MergeEndRecord()

Parameters
None

MergeExecute

Function: Executes a merge.

Equivalent Command: Tools-Merge-Merge

Syntax
MergeExecute(
PrimaryFile:string;
SecondaryFile:string
)

Parameters
PrimaryFile: Specifies the primary merge file. Enclose filename in quotes (include path if necessary).

SecondaryFile: Specifies the secondary merge file (optional). Enclose filename in quotes (include path if necessary).

MergeFieldDlg

Function: Calls the Insert Merge Code (field) dialog box.

Equivalent Command: Tools-Merge-Field

Syntax
MergeFieldDlg()

Parameters
None

MergeFilesDlg

Function: Calls the Insert Merge dialog box.

Equivalent Command: Tools-Merge-Merge

Syntax
MergeFilesDlg()

Parameters
None

MergeInputDlg

Function: Calls the Insert Merge Code (input) dialog box.

Equivalent Command: Tools-Merge-Input

Syntax
MergeInputDlg()

Parameters
None

MergeInsertCode

Function: Inserts the specified merge code, with optional parameters for the code.

Equivalent Command: Tools-Merge-Merge Codes

Syntax

MergeInsertCode
MergeCode:{
Assign!=0;
Bell!=1;
Break!=2;
Call!=3;
CancelOff!=4;
CancelOn!=5;
Case!=6;
CaseCall!=7;
ChainMacro!=8;
ChainPrimary!=9;
ChainSecondary!=10;
Char!=11;
Comment!=12;
Cton!=13;
Date!=14;
Document!=15;
Else!=16;
EndField!=17;
EndFor!=18;
EndIf!=19;
EndRecord!=20;
EndWhile!=21;
Field!=22;
For!=23;
Go!=25;
If!=26;
IfBlank!=27;
IfExists!=28;
IfNotBlank!=29;
Keyboard!=30;
Label!=31;
Local!=32;
Look!=33;
Mid!=34;
MergeCommand!=35;
NestMacro!=36;
NestPrimary!=37;
NestSecondary!=38;
Next!=39;
NextRecord!=40;
Ntoc!=41;
Process!=42;
OnCancel!=43;
OnError!=44;

```
                PageOff!=45;
                PageOn!=46;
                Print!=47;
                Prompt!=48;
                Quit!=49;
                Return!=50;
                ReturnCancel!=51;
                ReturnError!=52;
                Rewrite!=53;
                StepOff!=54;
                StepOn!=55;
                SubstPrimary!=56;
                SubstSecondary!=57;
                System!=58;
                Text!=59;
                Variable!=60;
                Wait!=61;
                While!=62;
                StatusPrompt!=63;
                Input!=64;
                Len!=65;
                FieldNames!=66;
                Stop!=67;
                Otherwise!=68
        };
        ParameterString:string
)
```

Parameters

MergeCode: Specifies the merge code to insert.

ParameterString: Specifies the parameter expression for the command. Include tildes if the command uses more than one parameter. Enclose the string in quotes.

MergeNextRecord

Function: Inserts a {NEXT RECORD} code.

Equivalent Command: Tools-Merge-Next Record

Syntax
MergeNextRecord()

Parameters
None

MergePageOff

Function: Inserts a {PAGE OFF} code.

Equivalent Command: Tools-Merge-Page Off

Syntax
MergePageOff()

Parameters
None

MergeVariableGet

Function: Retrieves the contents of a variable previously stored during a merge operation and places it into a macro variable.

Equivalent Command: Macro only.

Syntax

```
MergeVariableGet(
        MacroVariable:variable;
        MergeVariable:string
)
```

Parameters

MacroVariable: Specifies the name of the macro variable that will hold the contents of the merge variable.

MergeVariable: Specifies the name of the merge variable to access. The variable will be empty if it has not been previously assigned during a merge operation. Enclose the string in quotes.

Merges allow for "local" variables, which are cleared after the merge finishes. You cannot access the contents of these variables using the **MergeVariableGet** command.

MergeVariableSet

Function: Creates a merge variable using the contents of a macro string variable.

Equivalent Command: Macro only.

Syntax

```
MergeVariableSet(
        MergeVariable:string;
        MacroVariable:string
)
```

Parameters

MergeVariable: Specifies the name of the merge variable to create. Enclose the string in quotes.

MacroVariable: Specifies the string to copy to the merge variable. The string must be less than 128 characters. Enclose the string in quotes. You can also use a WPWin macro variable that contains a string.

NoteNext

Function: Views the next endnote or footnote, if any.

Equivalent Command: [Next] (in Footnote/Endnote Editor window)

Syntax
NoteNext()

Parameters
None

NoteNumber

Function: Inserts a note number code.

Equivalent Command: [Note Number] (in Footnote/Endnote Editor window)

Syntax
NoteNumber()

Parameters
None

NotePrevious

Function: Views the previous endnote or footnote, if any.

Equivalent Command: [Previous] (in Footnote/Endnote Editor window)

Syntax
NotePrevious()

Parameters
None

NumberFormat

Function: Specifies characters used for decimal alignment and thousands separator.

Equivalent Command: Layout-Line-Special Codes-
[Decimal Align] / [Thousands Separator]

Syntax
NumberFormat(
ThousandsSeparator:string;
DecimalAlignChar:string
)

Parameters
ThousandsSeparator: Specifies the character to use for the thousands separator.

DecimalAlignChar: Specifies the character to use for the alignment character.

OutlineCopy

Function: Copies the current outline family.

Equivalent Command: Tools-Outline-Copy Family

Syntax
OutlineCopy()

Parameters
None

OutlineDefine

Function: Defines an outline format.

Equivalent Command: Tools-Outline-Define

Syntax OutlineDefine(
Style:string;
InsertNumber:{
No!=0;
Yes!=1
};
AutoAdjust:{
No!=0;
Yes!=1
};
Level1:string;
Level2:string;
AttachLevel1:{
No!=0;
Yes!=1
};
Level3:string;
AttachLevel2:{
No!=0;
Yes!=1
};
...
Level8:string;
AttachLevel7:{
No!=0;
Yes!=1
};
StartingNumber:string
)

Parameters
Style: Specifies the style name of the outline, if any. Enclose the string in quotes.

InsertNumber: Specifies whether pressing the **[Enter]** key inserts a new paragraph number.

AutoAdjust: Specifies whether WPWin should automatically insert a new number at the last level used.

Level**x**: Specifies how level 1 through 8 numbers will appear. Enclose the string in quotes.

AttachLevel**x**: Specifies if current level is attached to the previous level.

StartingNumber: Specifies the starting number for outline numbering. Enclose the string in quotes.

OutlineDelete

Function: Deletes the current outline family.

Equivalent Command: Tools-Outline-Delete Family

Syntax
OutlineDelete()

Parameters
None

OutlineMove

Function: Moves the current outline family.

Equivalent Command: Tools-Outline-Move Family

Syntax
OutlineMove()

Parameters
None

OutlineOff

Function: Turns outlining mode off.

Equivalent Command: Tools-Outline-Outline Off

Syntax
OutlineOff()

Parameters
None

OutlineOn

Function: Turns outlining mode on.

Equivalent Command: Tools-Outline-Outline On

Syntax
OutlineOn()

Parameters
None

OverstrikeCreate

Function: Creates an overstrike character.

Equivalent Command: Font-Overstrike-Create

Syntax
OverstrikeCreate(
	Chars:string
)

Parameters
Chars: Specifies two or more characters to use for the overstrike.

OverstrikeCreateDlg

Function: Calls the Create/Edit Overstrike dialog.

Equivalent Command: Font-Overstrike-Create

Syntax
OverstrikeCreateDlg()

Parameters
None

OverstrikeEdit

Function: Edits an overstrike character.

Equivalent Command: Font-Overstrike-Edit

Syntax
OverstrikeEdit(
	Chars:string
)

Parameters
Chars: Specifies two or more new characters to use for the overstrike.

OverstrikeEditDlg

Function: Calls the Create/Edit Overstrike dialog.

Equivalent Command: Font-Overstrike-Edit

Syntax
OverstrikeEditDlg()

Parameters
None

PageCenter

Function: Centers the current page when printed.

Equivalent Command: Layout-Page-Center Page

Syntax

```
PageCenter(
        Set:{
                Off!=0;
                On!=1
        }
)
```

Parameters

Set: Specifies whether page centering is turned on or off (affects only the current page).

PageMargins

Function: Sets the top, right, left, and bottom margins.

Equivalent Command: Layout-Margins

Syntax
PageMargins(
    Left:wpunits;
    Right:wpunits;
    Top;wpunits;
    Bottom;wpunits
)

Parameters
The four parameters specify the left, right, top, and bottom margins, respectively. Value: any valid wpunits measurement.

PageMarginsDlg

Function: Calls the Margins dialog box.

Equivalent Command: Layout-Margins

Syntax
PageMarginsDlg()

Parameters
None

PageNumbering

Function: Turns page numbering on or off, and sets the style and position of page numbering.

Equivalent Command: Layout-Page-Numbering

Syntax

```
PageNumbering(
        Where:{
                None!=0;
                TopLeft!=1;
                TopCenter!=2;
                TopRight!=3;
                TopAlternate!=4;
                BottomLeft!=5;
                BottomCenter!=6;
                BottomRight!=7;
                BottomAlternate!=8
        };
        Text:string;
        NewNumber:number;
        Style:{
                Arabic!=0;
                LowerCaseRoman!=1;
                UpperCaseRoman!=2
        }
)
```

Parameters

Where: Specifies the position of page numbering (or turns page numbering off when **None!** is selected).

Text: Specifies accompanying text to use with the ^B page number code.

NewNumber: Specifies a new starting number, between 0 and 32767.

Style: Specifies the numbering style as Arabic, lower-case Roman, or upper-case Roman.

PageNumberingDlg

Function: Calls the Page Numbering dialog box.

Equivalent Command: Layout-Page-Page Numbering

Syntax
PageNumberingDlg()

Parameters
None

PageNumberInsert

Function: Inserts a page number code.

Equivalent Command: Layout-Page-Numbering-
[Insert Page Number]

Syntax

```
PageNumberInsert(
        WithSyle:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

WithStyle: Specifies whether the text in the Accompanying Text box (in the Page Numbering dialog) is included with the page number code.

PageSuppress

Function: Suppresses any or all of the following for the current page only: header, footer, page number.

Equivalent Command: Layout-Page-Suppress

Syntax

```
PageSuppress(
        HeaderA:{
                No!=0;
                Yes!=1;
        };
        HeaderB:{
                No!=0;
                Yes!=1;
        };
        FooterA:{
                No!=0;
                Yes!=1;
        };
        FooterB:{
                No!=0;
                Yes!=1;
        };
        PageNumber:{
                No!=0;
                Yes!=1;
        };
        PageToBottom:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

HeaderA/B: Specifies if Header A (or B) is suppressed.

FooterA/B: Specifies if Footer A (or B) is suppressed.

PageNumber: Specifies if page numbering is suppressed.

PageToBottom: Specifies if page number is printed at the bottom of the current page, even if page numbering is suppressed.

PageSuppressDlg

Function: Calls the Suppress dialog box.

Equivalent Command: Layout-Page-Suppress

Syntax
PageSuppressDlg()

Parameters
None

PageWidowOrphan

Function: Turns widow/orphan protection on or off.

Equivalent Command: Layout-Page-Widow/Orphan

Syntax

```
PageWidowOrphan(
        Protect:{
                Off!=0;
                On!=1
        }
)
```

Parameters

Protect: Specifies whether widow/orphan protection is on or off.

PaperSizeAdd

Function: Creates a new paper/size form.

Equivalent Command: Layout-Page-Paper Size-[Add]

Syntax

```
PaperSizeAdd(
        Type:{
                AllOthers!=0;
                Standard!=1;
                Bond!=2;
                Letterhead!=3;
                Labels!=4;
                Envelope!=5;
                Transparency!=6;
                Cardstock!=7;
                Other!=8
        };
        TypeName:string;
        FormLength:wpunits;
        Width:wpunits;
        TopAdj:wpunits;
        SideAdj:wpunits;
        FontOrient:{
                Port!=1;
                Land!=2;
                Both!=3
        };
        Loaded:{
                No!=0;
                Yes!=1
        };
        Location:number;
        BindEdge:{
                Long!=0;
                Short!=1
        };
        Duplex:{
                No!=0;
                Yes!=1
        };
        LabRows:number;
        LabCols:number;
        LabLeft:wpunits;
        LabTop:wpunits;
        LabWidth:wpunits;
        LabLength:wpunits;
        LabBetwnRows:wpunits;
        LabBetwnCols:wpunits;
        LabMarLeft:wpunits;
        LabMarRight:wpunits;
        LabMarTop:wpunits;
```

LabMarBottom:wpunits
)

Parameters
Type: Specifies the type of form.

TypeName: Specifies the name of the form. Enclose the string in quotes.

FormLength: Specifies the length of the form. Value: any valid wpunits measurement.

Width: Specifies the width of the form. Value: any valid wpunits measurement.

TopAdj: Specifies adjustment measurement from the top of the form. Value: any valid wpunits measurement.

SideAdj: Specifies adjustment measurement from the left side of the form. Value: any valid wpunits measurement.

FontOrient: Specifies the orientation of the page.

Loaded: Specifies whether paper pre-loaded in the paper bin before printing.

Location: Specifies the paper bin number. Use a value from 1 to 31 to print using a specified bin; use the value 32 to print using manual feed.

BindEdge: Specifies whether binding will be applied to the long or short edge of the paper.

Duplex: Specifies whether printing should be on both sides of the page.

LabRows: Specifies the number of label rows. Use -1 if you're not specifying labels; you can omit all other label specification parameters.

LabCols: Specifies the number of label columns.

LabLeft: Specifies the distance from the left edge of the paper to the first label. Value: any valid wpunits measurement.

LabTop: Specifies the distance from the top edge of the paper to the first label. Value: any valid wpunits measurement.

LabWidth: Specifies the width of each label. Value: any valid wpunits measurement.

LabLength: Specifies the height of each label. Value: any valid wpunits measurement.

LabBetwnRows: Specifies the distance between label rows. Value: any valid wpunits measurement.

LabBetwnCols: Specifies the distance between label columns. Value: any valid wpunits measurement.

LabMarLeft: Specifies the left margin for individual labels. Value: any valid wpunits measurement.

LabMarRight: Specifies the right margin for individual labels. Value: any valid wpunits measurement.

LabMarTop: Specifies the top margin for individual labels. Value: any valid wpunits measurement.

LabMarBottom: Specifies the bottom margin for individual labels. Value: any valid wpunits measurement.

PaperSizeDelete

Function: Deletes a paper size/type form. As the names for paper size/types can be duplicated, you must specify the parameters of the form along with its name.

Equivalent Command: Layout-Page-Paper Size-[Delete]

Syntax
(See PaperSizeAdd)
Parameters

(See PaperSizeAdd)

PaperSizeDlg

Function: Calls the Paper Size dialog box.

Equivalent Command: Layout-Page-Paper Size

Syntax
PaperSizeDlg()

Parameters
None

PaperSizeSelect

Function: Selects a paper size/type form. As the names for paper size/types can be duplicated, you must specify the parameters of the form along with its name.

Equivalent Command: Layout-Page-Paper Size-[Select]

Syntax
(See PaperSizeAdd)

Parameters
(See PaperSizeAdd)

ParagraphDoubleIndent

Function: Formats paragraph with double indenting.

Equivalent Command: Layout-Paragraph-Double Indent

Syntax
ParagraphDoubleIndent()

Parameters
None

ParagraphHangingIndent

Function: Formats paragraph with hanging indent.

Equivalent Command: Layout-Paragraph-Hanging Indent

Syntax
ParagraphHangingIndent()

Parameters
None

ParagraphIndent

Function: Formats paragraph with indent.

Equivalent Command: Layout-Paragraph-Indent

Syntax
ParagraphIndent()

Parameters
None

ParagraphNumber

Function: Inserts paragraph numbering code.

Equivalent Command: Tools-Outline-Paragraph Number

Syntax

```
ParagraphNumber(
        Level:number;
        InsertType:{
                Auto!=0;
                Fixed!=1
        }
)
```

Parameters

Level: Specifies the level (1 through 8) of the outline.

InsertType: Specifies whether paragraph level numbering is calculated automatically, or is fixed using the Level value.

ParagraphNumberDefDlg

Function: Calls the Define Paragraph Numbering dialog.

Equivalent Command: Tools-Outline-Define

Syntax
ParagraphNumberDefDlg()

Parameters
None

ParagraphNumberingDlg

Function: Calls the Paragraph Numbering dialog box.

Equivalent Command: Tools-Outline-Paragraph Number

Syntax
ParagraphNumberingDlg()

Parameters
None

PauseKey

Function: Pauses the macro until a specified key is pressed.

Equivalent Command: Macro only.

Syntax

```
PauseKey(
        Key:{
                Enter!=0;
                Cancel!=1;
                Close!=2;
                Character!=3
        };
        Character:string
)
```

Parameters

Key: Specifies the key or action used to release the macro from its paused state. **Cancel!** applies to both the Cancel key (usually **[Esc]**) and Cancel buttons. **Close!** applies to a Close button or command. If this parameter is omitted, **PauseKey** defaults to **Enter!**.

Character: When Key: is set to **Character!**, specifies the alphabetic or numeric key used to terminate the pause. Use only one character (only the first character is recognized); enclose the string in quotes.

This command is not equivalent to the Macro-Pause command, which is used to pause the macro during recording. See Pause for more information.

PosDirection

Function: Moves the insertion point in the specified direction or to the specified position.

Equivalent Command: Various.

Syntax
PosCellDown()
PosCellNext()
PosCellPrevious()
PosCellUp()
PosCharNext()
PosCharPrevious()
PosColumnNext()
PosColumnPrevious()
PosDocBottom()
PosDocTop()
PosDocVeryTop
PosLineBegin()
PosLineDown()
PosLineEnd()
PosLineUp()
PosLineVeryBegin()
PosLineVeryEnd()
PosPageBottom()
PosPageNext()
PosPagePrevious()
PosPageTop()
PosParagraphNext()
PosParagraphPrevious()
PosScreenDown()
PosScreenLeft()
PosScreenRight()
PosScreenUp()
PosTableBegin()
PosTableColumnBottom()
PosTableColumnTop()
PosTableEnd()
PosTableRowBegin()
PosTableRowEnd()
PosWordNext()
PosWordPrevious()

Parameters
None

PosGoTo

Function: Moves the insertion point to a specific place or page.

Equivalent Command: Edit-Go to

Syntax

```
PosGoTo(
        Where:{
                BeginningOfSelection!=0;
                ReselectText!=1;
                TopOfPage!=2;
                BottomOfPage!=3;
                TopOfColumn!=4;
                BottomOfColumn!=5;
                PreviousColumn!=6;
                NextColumn!=7;
                FirstColumn!=8;
                LastColumn!=9;
                TopOfCell!=10;
                BottomOfCell!=11;
                FirstCell!=12;
                LastCell!=13;
                LastPosition!=14
        }
        Page:number;
        Cell:string
)
```

Parameters

Where: Specifies where you want to move the insertion point. Note that some values are not valid at all times; for example, BottomOfCell will be ignored if the cursor is not currently in a table.

Page: Specifies a particular page in the document. Valid values are from 1 to the maximum number of pages in the document. WordPerfect will ignore the command if you specify a page that is out of range.

Cell: Specifies a particular cell in the current table. Value is a string; enclose in quotes.

The parameters of PosGoTo are exclusive of one another. Use only one of the three parameters, as needed. For example, when specifying a page number, do not also specify a cell or location on the page.

PosGoToDlg

Function: Calls the Go To dialog box.

Equivalent Command: Edit-Go to

Syntax
PosGoToDlg()

Parameters
None

PrefBackup

Function: Sets backup preferences.

Equivalent Command: File-Preferences-Backup

Syntax

```
PrefBackup(
        TimedBackup:{
                Off!=0;
                On!=1
        };
        MinutesBetween:number;
        OriginalDocBackup:{
                Off!=0;
                On!=1
        }
)
```

Parameters

TimedBackup: Specifies whether timed backup is on or off.

MinutesBetween: Specifies minutes between timed backups. Valid values are between 1 and 65535.

OriginalDocBackup: Specifies whether the document backup feature is on or off.

Use PrefSave to save changes made with this command.

PrefBackupDlg

Function: Calls the Backup dialog box.

Equivalent Command: File-Preferences-Backup

Syntax
PrefBackupDlg()

Parameters
None

PrefBeep

Function: Sets error beep preferences.

Equivalent Command: File-Preferences-Environment

Syntax

```
PrefBeep(
        Error:{
                Off!=0;
                On!=1
        };
        Hyphenation:{
                Off!=0;
                On!=1
        };
        SearchFailure:{
                Off!=0;
                On!=1
        }
)
```

Parameters

Error: Specifies whether beep is on or off for errors.

Hyphenation: Specifies whether beep is on or off for hyphenation prompts,

SearchFailure: Specifies whether beep is on or off for failed searches.

Use PrefSave to save changes made with this command.

PrefDateFormat

Function: Sets date format preferences.

Equivalent Command: File-Preferences-Date Format
Syntax
PrefDateFormat(
FormatString:string
)

Parameters
FormatString: Specifies the date format. Enclose the string in quotes. The following codes are used to include date and time information:

Character	Meaning
1	Day of the month
2	Month (number)
3	Month (spelled out)
4	Year (four digits)
5	Year (last two digits)
6	Day of the week (spelled out)
7	Hour (24-hour clock)
8	Hour (12-hour clock)
9	Minute
0	am/pm
$	Pad numbers less than 10 with space
%	Pad numbers less than 10 with zero; abbreviates day of week and month

Use PrefSave to save changes made with this command.

PrefDateFormatDlg

Function: Calls the Date/Time Format dialog box.

Equivalent Command: File-Preferences-Date Format

Syntax
PrefDateFormatDlg()

Parameters
None

PrefDisplayDlg

Function: Calls the Display Settings dialog box.

Equivalent Command: File-Preferences-Display

Syntax PrefDisplayDlg()

Parameters
None

PrefDisplaySet

Function: Alters display setting preferences.

Equivalent Command: File-Preferences-Display

Syntax

```
PrefDisplaySet(
        TextInSystemColors:{
                No!=0;
                Yes!=1
        };
        MonochromeGraphics:{
                No!=0;
                Yes!=1
        };
        AutoFormat:{
                No!=0;
                Yes!=1
        };
        ColumnsSideBySide:{
                No!=0;
                Yes!=1
        };
        DispMergeCodes:{
                No!=0;
                Yes!=1
        };
        SculpturedDialogs:{
                No!=0;
                Yes!=1
        };
        VerticalScrollBar:{
                No!=0;
                Yes!=1
        };
        HorizontalScrollBar:{
                No!=0;
                Yes!=1;

                AutoZoom!=2
        };
        HardReturnChar:string;
        DisplayUnits:{
                InchesQuote!=0;
                InchesI!=1;
                Centimeters!=2;
                Points!=3;
                WPUs!=4
        };
        StatusBarUnits:{
                InchesQuote!=0;
                InchesI!=1;
```

```
            Centimeters!=2;
            Points!=3;
            WPUs!=4
        };

        RevealCodesSize:number
        }
)
```

Parameters

TextInSystemColors: Specifies whether text is shown in Windows system colors.

MonochromeGraphics: Specifies whether graphics are shown in black and white.

AutoFormat: Specifies whether the screen is automatically reformatted while in Draft mode.

ColumnsSideBySide: Specifies whether columns are displayed side-by- side.

DispMergeCodes: Specifies whether merge codes are displayed.

SculpturedDialogs: Specifies whether dialog boxes are shown with 3-D sculpturing.

VerticalScrollBar: Specifies whether vertical scroll bar is displayed.

HorizontalScrollBar: Specifies whether horizontal scroll bar is displayed.

HardReturnChar: Specifies hard return character. Enclose string in quotes.

DisplayUnits: Specifies units of measure for dialog boxes, etc.

StatusBarUnits: Specifies units of measure for the status bar.

RevealCodesSize:  Specifies the Reveal Codes size, as a percentage. Valid values: 1 to 99.

Use PrefSave to save changes made with this command.

PrefDocSummary

Function: Sets document summary preferences

Equivalent Command: File-Preferences-Document Summary

Syntax

```
PrefDocSummary(
        SubjectSearchText:string;
        DefaultDocType:string;
        CreateOnExit:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

SubjectSearchText: Specifies text that precedes subject matter of documents.

DefaultDocType: Specifies the text that will automatically appear in the Descriptive Type input box when a document summary is created.

CreateOnExit: Specifies whether the Document Summary dialog box always appears when a document is closed or first saved.

Use PrefSave to save changes made with this command.

PrefDocSummaryDlg

Function: Calls the Document Summary dialog box.

Equivalent Command: File-Preferences-Document Summary

Syntax
PrefDocSummaryDlg()

Parameters
None

PrefDraftColors

Function: Sets colors for Draft mode.

Equivalent Command: File-Preferences-Display-[Draft Mode Colors]

Syntax

```
PrefDraftColors(
        FontAttribute:{
                Normal!=0;
                Blocked!=1;
                Underline!=2;
                Strikeout!=3;
                Bold!=4;
                DoubleUnderline!=5;
                Redline!=6;
                Shadow!=7;
                Italics!=8;
                SmallCaps!=9;
                Outline!=10;
                Subscript!=11;
                Superscript!=12;
                Fine!=13;
                Small!=14;
                Large!=15;
                VeryLarge!=16;
                ExtraLarge!=17;
                BoldUnderline!=18;
                Combinations!=19
        };
        Foreground:{
                White!=0;
                DkGray!=1;
                LtRed!=2;
                LtYellow!=3;
                LtGreen!=4;
                LtCyan!=5;
                LtBlue!=6;
                LtMagenta!=7;
                LtGray!=8;
                Black!=9;
                DkRed!=10;
                DkYellow!=11;
                DkGreen!=12;
                DkCyan!=13;
                DkBlue!=14;
                DkMagenta!=15
        };Background:{
                White!=0;
                DkGray!=1;
                DkCyan!=2;
                DkMagenta!=3;
                LtGray!=4;
```

```
                Black!=5;
                DkBlue!=6;
                LtRed!=7
        }
)
```

Parameters

FontAttribute: Specifies the attribute to change.

Foreground: Specifies the foreground (text) color.

Background: Specifies the background color.

Use PrefSave to save changes made with this command.

PrefDraftColorsDlg

Function: Calls the Draft Mode Colors dialog box.

Equivalent Command: File-Preferences-Display-
[Draft Mode Colors]

Syntax
PrefDraftColorsDlg()

Parameters
None

PrefEnvSettings

Function: Sets environment preferences.

Equivalent Command: File-Preferences-Environment

Syntax

```
PrefEnvSettings(
        AutoCodePlacement:{
                Off!=0;
                On!=1
        };
        ConfirmCodeDeletion:{
                No!=0;
                Yes!=1
        };
        FastSave:{
                Off!=0;
                On!=1
        };
        AllowUndo:{
                No!=0;
                Yes!=1
        };
        FormatForPrinter:{
                No!=0;
                Yes!=1
        };
        GraphicsBumping:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

AutoCodePlacement: Specifies whether AutoCode Placement is on or off.

ConfirmCodeDeletion: Specifies whether WPWin warns the user when deleting codes (when not in Reveal Codes).

FastSave: Specifies whether files are re-saved in Fast Save format.

AllowUndo: Specifies whether Undo is allowed.

FormatForPrinter: Specifies whether retrieved documents are reformatted for currently selected printer.

GraphicsBumping: Specifies whether a page anchor graphics can be "bumped" to the next page if new text is added before the graphics code.

Use PrefSave to save changes made with this command.

PrefEnvSettingsDlg

Function: Calls the Environment Preferences dialog.

Equivalent Command: File-Preferences-Environment

Syntax
PrefEnvSettingsDlg()

Parameters
None

PrefEquation

Function: Sets equation preferences.

Equivalent Command: File-Preferences-Equation

Syntax

```
PrefEquation(
        PrintAsGraphics:{
                Text!=0;
                Graphics!=1
        };
        FontSize:points;
        HorizontalAlignment:{
                Left!=0;
                Right!=1;
                Center!=2
        };
        VerticalAlignment:{
                Top!=0;
                Center!=1;
                Bottom!=2
        };
        Keyboard:filename
)
```

Parameters

PrintAsGraphics: Specifies whether equations are to be printed as graphics.

FontSize: Specifies default font size for equation text. Value: any valid font size, in points.

HorizontalAlignment: Specifies the horizontal alignment of the equation inside its box.

VerticalAlignment: Specifies the vertical alignment of the equation inside its box.

Keyboard: Specifies the keyboard to use while in the Equation Editor. Enclose filename in quotes (include path if necessary).

Use PrefSave to save changes made with this command.

PrefEquationDlg

Function: Calls the Equation Settings Preferences dialog box.

Equivalent Command: File-Preferences-Equations

Syntax
PrefEquationDlg()

Parameters
None

PrefHyphenation

Function: Sets the hyphenation preferences.

Equivalent Command: File-Preferences-Environment

Syntax

```
PrefHyphenation(
        Prompt:{
                Never!=0;
                WhenRequired!=1;
                Always!=2
        };
        HyphType:{
                External!=0;
                Internal!=1
        }
)
```

Parameters

Prompt: Specifies when to prompt for word hyphenation.

HyphType: Specifies the source of the hyphenation dictionary (**Internal!** uses an internal hyphenation algorithm).

Use PrefSave to save changes made with this command.

PrefInitialCodes

Function: Sets initial codes preferences.

Equivalent Command: File-Preferences-Initial Codes

Syntax
PrefInitialCodes()

Parameters
None

- WordPerfect opens the Initial Codes window after invoking the **PrefInitialCodes** command. Format commands you issue place codes in this window.
- Use the Close command to close the Initial Codes window when you are done setting the format.
- Use the PrefSave command to save the Initial Codes changes. Otherwise, when you leave WPWin the Initial Codes changes are lost.

PrefLocationDlg

Function: Calls the Location of Files dialog box.

Equivalent Command: File-Preferences-Location of Files

Syntax
PrefLocationDlg()

Parameters
None

PrefLocationOfFiles

Function: Specifies the path for critical WPWin files.

Equivalent Command: File-Preferences-Location of Files

Syntax

```
PrefLocationOfFiles(
        BackupFiles:string;
        Documents:string;
        GraphicFiles:string;
        PrinterFiles:string;
        Spreadsheets:string;
        MacroFiles:string;
        StyleFiles:string;
        StyleLibraryName:string;
        MainDictFiles:string;
        SupDictFiles:string;
        UpdateQuickList:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

BackupFiles: Specifies the timed backup file path.

Documents: Specifies the document files path.

GraphicFiles: Specifies the graphic files path.

PrinterFiles: Specifies the printer files path.

Spreadsheets: Specifies the spreadsheet files path.

MacroFiles: Specifies the macro files path.

StyleFiles: Specifies the style files path.

StyleLibraryName: Specify the name and path of the style library.

MainDictFiles: Specifies the main dictionary files path.

SupDictFiles: Specifies the supplemental dictionary files path.

UpdateQuickList: Specifies whether dialog box Quick Lists are updated with new entries specified in Location of Files.

- With the exception of UpdateQuickList, all parameters are strings, and should be enclosed in quotes.
- Use PrefSave to save changes made with this command.

PrefMenu

Function: Sets menu preferences.

Equivalent Command: File-Preferences-Environment

Syntax

```
PrefMenu(
        DisplayShortcuts:{
                No!=0;
                Yes!=1
        };
        DisplayLastOpened:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

DisplayShortcuts: Specifies if shortcut (accelerator) keys are displayed in menus.

DisplayLastOpened: Specifies if the last four opened files are shown at the bottom of the File menu.

Use PrefSave to save changes made with this command.

PrefMergeDelimit

Function: Sets preferences for default field and record delimiters when using DOS text files for secondary merge documents.

Equivalent Command: File-Preferences-Merge

Syntax

```
PrefMergeDelimit(
        FieldBegin:string;
        FieldEnd:string;
        RecordBegin:string;
        RecordEnd:string
)
```

Parameters

FieldBegin: Specifies the character to delimit the beginning of fields.

FieldEnd: Specifies the character to delimit the end of fields.

RecordBegin: Specifies the character to delimit the beginning records.

RecordEnd: Specifies the characters to delimit the end of records.

Use PrefSave to save changes made with this command.

PrefMergeDlg

Function: Calls the Merge Preferences dialog box.

Equivalent Command: File-Preferences-Merge

Syntax
PrefMergeDlg()

Parameters
None

PrefPrintCopies

Function: Sets default number of copies to print.

Equivalent Command: File-Preferences-Print

Syntax

```
PrefPrintCopies(
        NumberOfCopies:number;
        GeneratedBy:{
                WordPerfect!=0;
                Printer!=1
        }
)
```

Parameters

NumberOfCopies: Specifies the number of copies to print of each page of the document. Valid values are 1 to 16383.

GeneratedBy: Specifies if WordPerfect or your printer will generate the multiple copies.

Use PrefSave to save changes made with this command.

PrefPrintDoc

Function: Sets document printing preferences.

Equivalent Command: File-Preferences-Print

Syntax

```
PrefPrintDoc(
        BindingOffset:wpunits;
        GraphicsQuality:{
                DoNotPrint!=0;
                Draft!=1;
                Medium!=2;
                High!=3
        };
        TextQuality:{
                DoNotPrint!=0;
                Draft!=1;
                Medium!=2;
                High!=3
        }
)
```

Parameters

BindingOffset: Specifies the binding offset, if any. Value: any valid wpunits measurement.

GraphicsQuality: Specifies the graphics printing quality (resolution). TextQuality: Specifies the text printing quality (resolution).

Use PrefSave to save changes made with this command.

PrefPrintRedline

Function: Sets redline printing preferences.

Equivalent Command: File-Preferences-Print

Syntax

```
PrefPrintRedline(
        Method:{
                PrinterDependent!=0;
                Left!=1;
                Alternating!=2
        };
        Character:string
)
```

Parameters

Method: Specifies if redline method is printer dependent, or appears as a column of characters in the left or alternating margins.

Character: If Method: is set to **Left!** or **Alternating!**, specifies the single character to use to denote redlining.

Use PrefSave to save changes made with this command.

PrefPrintSettingsDlg

Function: Calls the Print Settings dialog box.

Equivalent Command: File-Preferences-Print

Syntax
PrefPrintSettingsDlg()

Parameters
None

PrefPrintSizeRatio

Function: Changes the preset font size printing ratios.

Equivalent Command: File-Preferences-Print

Syntax

```
PrefPrintSizeRatio(
        Fine:number;
        Small:number;
        Large:number;
        VeryLarge:number;
        ExtraLarge:number;
        SuperSubscript:number
)
```

Parameters

All parametersare integer (while number) values representing a percentage of the current base font. Valid values are from 1 to 999.

Use PrefSave to save changes made with this command.

PrefPrintWindowsDriverSet

Function: Sets the preference for Windows printer driver.

Equivalent Command: File-Preferences-Print

Syntax

```
PrefPrintWindowsDriverSet(
        FastGraphics:{
                No!=0;
                Yes!=1
         }
)
```

Parameters

FastGraphics: Specifies whether fast graphics printing for Windows drivers is enabled or disabled.

PrefRevealCodeColors

Function: Sets colors for Reveal Codes.

Equivalent Command: File-Preferences-Display-
[Reveal Codes Colors]

Syntax

```
PrefRevealCodeColors(
        FontAttribute:{
                Text!=0;
                Codes!=1;
                Cursor!=2
        };
        Foreground:{
                White!=0;
                DkGray!=1;
                LtRed!=2;
                LtYellow!=3;
                LtGreen!=4;
                LtCyan!=5;
                LtBlue!=6;
                LtMagenta!=7;
                LtGray!=8;
                Black!=9;
                DkRed!=10;
                DkYellow!=11;
                DkGreen!=12;
                DkCyan!=13;
                DkBlue!=14;
                DkMagenta!=15
        };
        Background:{
                White!=0;
                DkGray!=1;
                DkCyan!=2;
                DkMagenta!=3;
                LtGray!=4;
                Black!=5;
                DkBlue!=6;
                LtRed!=7
        } )
```

Parameters

FontAttribute: Specifies the attribute to change.

Foreground: Specifies the foreground (text) color.

Background: Specifies the background color.

Use PrefSave to save changes made with this command.

PrefRevealCodeColorsDlg

Function: Calls the Reveal Codes Colors dialog box.

Equivalent Command: File-Preferences-Display-
[Reveal Codes Colors]

Syntax
PrefRevealCodeColorsDlg()

Parameters
None

PrefRuler

Function: Sets ruler preferences.

Equivalent Command: File-Preferences-Environment

Syntax

```
PrefRuler(
        TabsToGrid:{
                No!=0;
                Yes!=1
        };
        ShowRuler:{
                No!=0;
                Yes!=1
        };
        RulerButtonsOnTop:{
                No!=0;
                Yes!=1
        };
        AutoRulerDisplay:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

TabsToGrid: Specifies whether tabs are aligned to the ruler grid.

ShowRuler: Specifies whether ruler guides appear when moving tabs and margin controls.

RulerButtonsOnTop: Specifies whether ruler buttons are on top (bottom is default).

AutoRulerDisplay: Specifies whether the ruler appears for each new or opened document.

Use PrefSave to save changes made with this command.

PrefSave

Function: Saves changes made to preference settings.

Equivalent Command: Macro only.

Syntax
PrefSave()

Parameters
None

PrefToA

Function: Sets table of authorities preferences.

Equivalent Command: File-Preferences-Table of Authorities

Syntax

```
PrefToA(
        DotLeaders:{
                Off!=0;
                On!=1
        };
        Underlining:{
                NotAllowed!=0;
                Allowed!=1
        };
        BlankLineBetween:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

DotLeaders: Specifies whether dot leaders are used for ToA references.

Underlining: Specifies whether underlining is allowed in ToA.

BlankLineBetween: Specifies whether blank line is automatically inserted between ToA entries.

Use PrefSave to save changes made with this command.

PrefToADlg

Function: Calls the Preferences: Table of Authorities dialog.

Equivalent Command: File-Preferences-Table of Authorities

Syntax
PrefToADlg()

Parameters
None

PrefZoom 

Function: Sets the default zoom preference.

Equivalent Command: File-Preferences-Display-[Zoom]

Syntax
PrefZoom(
Percent:number
)

Parameters
Percent: Specifies the zoom percentage. Valid values: 50 to 400.


Use PrefSave to save changes made with this command.

PrefZoomDlg 

Function: Calls the Zoom Preferences dialog box.

Equivalent Command: File-Preferences-Display-[Zoom]

Syntax
PrefZoomDlg()

Parameters
None

PrintDlg

Function: Calls thePrint dialog box.

Equivalent Command: File-Print

Syntax
PrintDlg()

Parameters
None

PrintDoc

Function: Prints a document saved on disk.

Equivalent Command: File-Print-(Document on Disk)

Syntax

```
PrintDoc(
        DocumentName:string;
        Range:string;
        OddEven:{
                None!=0;
                Odd!=1;
                Even!=2;
                LogicalOdd!=3;
                LogicalEven!=4  };
        DocumentSummary:{
                No!=0;
                Yes!=1
        };
        NoWarnings:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

DocumentName: Specifies the file (with path, if necessary) to print. Enclose filename in quotes.

Range: Specifies the pages to print (enclose string in quotes). Omit this parameter to print the entire document.

OddEven: Specifies whether all, odd, or even pages are printed.

DocumentSummary: Specifies whether document summary (if any) is printed.

NoWarnings: Specifies whether print messages are suppressed or displayed during printing.

PrinterCommand

Function: Sends a command to the printer.

Equivalent Command: Layout-Typesetting-[Printer Command]

Syntax
PrinterCommand(
CommandString:string;
Filename:string
)

Parameters
CommandString: Specifies the command string to use. Enclose string in quotes.

Filename: Specifies a file containing printer commands. Enclose filename in quotes (include path if necessary).

PrinterCommandDlg

Function: Calls the Printer Command dialog box.

Equivalent Command: Layout-Typesetting-[Printer Command]

Syntax
PrinterCommandDlg()

Parameters
None

PrinterInitialize

Function: Initializes the printer.

Equivalent Command: Print-[Initialize Printer]

Syntax
PrinterInitialize()

Parameters
None

PrinterSelectDlg

Function: Calls Select Printer dialog box.

Equivalent Command: File-Select Printer

Syntax
PrinterSelectDlg()

Parameters
None

PrintFull

Function: Prints the entire current document.

Equivalent Command: File-Print-(Full Document)

Syntax
PrintFull()

Parameters
None

PrintMultiplePages

Function: Prints one or more pages from the current document.

Equivalent Command: File-Print-(Multiple Pages)

Syntax

```
PrintMultiplePages(
        Range:string;
        OddEven:{
                None!=0;
                Odd!=1;
                Even!=2;
                LogicalOdd!=3;
                LogicalEven!=4
        };
        DocumentSummary:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Range: Specifies the print range. Enclose string in quotes.

OddEven: Specifies whether all pages in are printed, or just odd or even pages.

DocumentSummary: Specifies whether document summary is printed.

PrintOptions

Function: Sets printing options.

Equivalent Command: File-Print

Syntax

```
PrintOptions(
        Copies:number;
        CopiesBy:{
                WordPerfect!=0;
                Printer!=1
        };
        Binding:wpunits;
        TextQuality:{
                DoNotPrint!=0;
                Draft!=1;
                Medium!=2;
                High!=3
        };
        GraphicsQuality:{
                DoNotPrint!=0;
                Draft!=1;
                Medium!=2;
                High!=3
        }
)
```

Parameters

Copies: Specifies the number of copies to print.

CopiesBy: Specifies whether WordPerfect or your printer generates multiple copies.

Binding: Specifies binding width, if any. Value: any valid wpunits measurement.

TextQuality: Specifies text quality (resolution).

GraphicsQuality: Specifies graphics quality (resolution).

PrintPage

Function: Prints the current page only.

Equivalent Command: File-Print-(Current Page)

Syntax
PrintPage()

Parameters
None

PrintPreview

Function: Selects Print Preview mode.

Equivalent Command: File-Print Preview.

Syntax
PrintPreview()

Parameters
None

PrintSelected

Function: Prints the currently selected text.

Equivalent Command: File-Print

Syntax
PrintSelected()

Parameters
None

Redisplay

Function: Refreshes the screen.

Equivalent Command: **[Ctrl+F3]**

Syntax
Redisplay()

Parameters
None

RevealCodes

Function: Shows or hides Reveal Codes.

Equivalent Command: View-Reveal Codes

Syntax

```
RevealCodes(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies the state of Reveal Codes.

RulerShow

Function: Shows or hides the ruler (current window only).

Equivalent Command: View-Ruler

Syntax

```
Rulershow(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies the state of the ruler.

ScrollScreenLeft

Function: Scrolls the screen left.

Equivalent Command: **[Ctrl+Page Up]**

Syntax
ScrollScreenLeft

Parameters
None

ScrollScreenRight

Function: Scrolls the screen right.

Equivalent Command: **[Ctrl+Page Down]**

Syntax
ScrollScreenRight()

Parameters
None

SearchDlg

Function: Calls the Search dialog box.

Equivalent Command: Edit-Search

Syntax
SearchDlg()

Parameters
None

SearchNext

Function: Repeats the most recent search forward through the document.

Equivalent Command: Edit-Search Next

Syntax
SearchNext()

Parameters
None

SearchPrevious

Function: Repeats the most recent search backward through the document.

Equivalent Command: Edit-Search Previous

Syntax
SearchPrevious()

Parameters
None

SearchReplace

Function: Searches the current document and replaces found text.
Equivalent Command: Edit-Replace
Syntax

```
SearchReplace(
        SearchString:string;
        SearchDirection:{
                Forward!=0;
                Backward!=1
        };
        ReplacementScope:{
                Extended!=0;
                DocOnly!=1
        };
        ReplacementString:string;
        ReplacementAction:{
                ReplaceAll!=1;
                ReplaceOne!=2
        }
)
```

Parameters
SearchString: Specifies the search string. Enclose string in quotes.

SearchDirection: Specifies the search direction.

ReplacementScope: Specifies the scope of the search.

ReplacementString: Specifies the replacement string. Enclose string in quotes.

ReplacementAction: Specifies whether search/replace should apply to the next occurrence of the search string, or for the rest of the document.

The search and replacement strings can contain formatting codes. These must be in the form of Embedded codes which cannot be directly entered into the macro document.

SearchReplaceDlg

Function: Calls the Search and Replace dialog box.

Equivalent Command: Edit-Replace

Syntax
SearchReplaceDlg()

Parameters
None

SearchText

Function: Searches the document for matching text.

Equivalent Command: Edit-Search

Syntax

```
SearchText(
        SearchString:string;
        SearchDirection:{
                Forward!=0;
                Backward!=1
        };
        SearchScope:{
                Extended!=0;
                DocOnly!=1;
                WithinBlock!=2
        }
)
```

Parameters

SearchString: Specifies the search string. Enclose string in quotes.

SearchDirection: Specifies the search direction.

SearchScope: Specifies the scope of the search.

The search and replacement strings can contain formatting codes. These must be in the form of Embedded codes which cannot be directly entered into the macro document.

SelectAction

Function: Selects text.
Equivalent Command: Various.
Syntax
SelectAll()
SelectCell()
SelectCellDown()
SelectCellDownArrow()
SelectCellLeft()
SelectCellRight()
SelectCellUp()
SelectCellUpArrow()
SelectCharNext()
SelectCharPrevious()
SelectColumn()
SelectColumnBottom()
SelectColumnNext()
SelectColumnPrevious()
SelectColumnTop()
SelectDocBottom()
SelectDocTop()
SelectDocVeryTop()
SelectLineBegin()
SelectLineDown()
SelectLineEnd()
SelectLineUp()
SelectLineVeryBegin()
SelectLineVeryEnd()
SelectPage()
SelectPageNext()
SelectPagePrevious()
SelectParagraph()
SelectParagraphNext()
SelectParagraphPrevious()
SelectRectangle()
SelectScreenDown()
SelectScreenLeft()
SelectScreenRight()
SelectScreenUp()
SelectSentence()
SelectSentenceNext()
SelectSentencePrevious()
SelectTable()
SelectTableColumn()
SelectTableColumnExtendLeft()
SelectTableColumnExtendRight()
SelectTableRow()
SelectWord()
SelectWordNext()
SelectWordPrevious()

SelectMode

Function: Turns Select mode on and off.

Equivalent Command: **[F8]**

Syntax

```
SelectMode(
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

State: Specifies whether Select mode is turned on or off.

ShortMenus

Function: Enables or disables short menus.

Equivalent Command: View-Short Menus

Syntax

```
ShortMenus(
        State:{
                Short!=0;
                Long!=1
        }
)
```

Parameters

State: Specifies whether menus are short (Short Menus on) or long (Short Menus off).

When using the May 1992 interim release, Short Menus must be explicitly enabled in the WPWP.INI file (using the ShortMenus= key, under the [Settings] heading), or the Short Menus command is not displayed in the View menu.

[Settings] Entry	Result
ShortMenus=0	Short Menus command not shown in View menu.
ShortMenus=1	Short Menus command shown in View menu.

Sort

Function: Sorts lines of text, paragraphs, merge records, or table rows.

Equivalent Command: Tools-Sort

Syntax

```
Sort(
        RecordType:{
                Line!=1;
                Paragraph!=2;
                MergeRec!=3;
                TableRow!=4
        };
        SortOrder:{
                Ascending!=1;
                Descending!=2;
                NoSort!=3
        };
        Selection:string;
        Key1Type:{
                Numeric!=0;
                Alpha!=1
        };
        Key1Level1:number;
        Key1Level2:number;
        Key1Level3:number;
        ...
        Key9Level1:number;
        Key9Level2:number;
        Key9Level3:number
)
```

Parameters

RecordType: Specifies the type of record to sort (line, paragraph, merge record, or table row).

SortOrder: Specifies the sort order.

Selection: Specifies search criteria, such as AND and OR. Enclose string in quotes.

Key**x**Type: Specifies the type of sort (alphabetic or numeric) for sort fields 1 through 9.

Key**x**Level1: Specifies the position of the first sort level. Use a positive value only.

Key**x**Level2: Specifies the position of the second sort level. Use a positive or negative value.

Key**x**Level3: Specifies the position of the third sort level. Use a positive or negative value.

- Use single digits for Key**x**Levels.
- Omit a level if you don't want to include it in the sort.

Sort Record Type	Level 1	Level 2	Level 3

Line	Field*	(not used)	Word
Paragraph	Line	Field*	Word
Merge Record	Field**	Line	Word
Table Row	Cell	Line	Word

* Tabs separate fields for lines and paragraphs

** {END FIELD} code and line break separate fields for merge records.

SortDlg

Function: Calls the Sort dialog box.

Equivalent Command: Tools-Sort

Syntax
SortDlg()

Parameters
None

Speller

Function: Calls the WordPerfect Speller utility.

Equivalent Command: Tools-Speller

Syntax
Speller()

Parameters
None

SpreadsheetImportDlg

Function: Calls the Import Spreadsheet dialog box.

Equivalent Command: Tools-Spreadsheet-Import

Syntax
SpreadsheetImportDlg()

Parameters
None

SpreadsheetImportLink

Function: Creates or edits a spreadsheet file link.

Equivalent Command: Tools-Spreadsheet-Create Link or
Tools-Spreadsheet-Edit Link

Syntax

```
SpreadsheetImportLink(
        Operation:{
                Import!=0;
                Create!=1;
                Edit!=2
        };
        Filename:string;
        RangeName:string;
        Type:{
                Text!=0;
                Table!=1
        }
)
```

Parameters

Operation: Specifies the linking operation (import, create, or edit).

Filename: Specifies the name of the spreadsheet file. Note: enclose filename in quotes (include path if necessary).

RangeName: Specifies the range of cells to link. Enclose the string in quotes.

Type: Specifies how the linked data will appear in the document.

SpreadsheetLinkCreateDlg

Function: Calls the Create Spreadsheet Link dialog box.

Equivalent Command: Tools-Spreadsheet-Create Link

Syntax
SpreadsheetLinkCreateDlg()

Parameters
None

SpreadsheetLinkEditDlg

Function: Calls the Edit Spreadsheet Link dialog box.

Equivalent Command: Tools-Spreadsheet-Edit Link

Syntax
SpreadsheetLinkEditDlg()

Parameters
None

SpreadsheetLinkOptions

Function: Sets options for spreadsheet links.

Equivalent Command: Tools-Spreadsheet-Link Options

Syntax

```
SpreadsheetLinkOptions(
        UpdateOnRetrieve:{
                No!=0;
                Yes!=1
        };
        ShowLinkCodes:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

UpdateOnRetrieve: Specifies whether spreadsheet links are updated when the document is opened or retrieved.

ShowLinkCodes: Specifies whether link codes will display on screen.

SpreadsheetLinkOptionsDlg

Function: Calls the Link Options dialog box.

Equivalent Command: Tools-Spreadsheet-Link Options

Syntax
SpreadsheetLinkOptionsDlg()

Parameters
None

SpreadsheetLinkUpdateAll

Function: Updates all spreadsheet links.

Equivalent Command: Tools-Spreadsheet-Update All Links

Syntax
SpreadsheetLinkUpdateAll()

Parameters
None

SpreadsheetLinkUpdateAllDlg

Function: Calls the Update Spreadsheet Links dialog box.

Equivalent Command: Tools-Spreadsheet-Update All Links

Syntax
SpreadsheetLinkUpdateAllDlg()

Parameters
None

Styles

Function: Turns a specified style on or off.

Equivalent Command: Layout-Styles

Syntax

```
Styles(
        Name:string;
        State:{
                Off!=0;
                On!=1
        }
)
```

Parameters

Name: Specifies the name of the style. Enclose the string in quotes.

State: Turns the style is to be turned on or off.

StylesDelete

Function: Deletes a specified style.

Equivalent Command: Layout-Styles-[Delete]

Syntax

```
StylesDelete(
        Name:string;
        Type:{
                Include!=0;
                Leave!=1;
                Definition!=2
        }
)
```

Parameters

Name: Specifies the name of the style to delete.

Type: Specifies the scope of the deletion.

Parameter	Corresponding WPWin Selection
Include!	Delete Format Codes
Leave!	Leave Format Codes
Definition!	Delete Definition Only

StylesDlg

Function: Calls the Styles dialog box.

Equivalent Command: Layout-Styles

Syntax
StylesDlg()

Parameters
None

StylesEdit

Function: Edits the specified style definition.

Equivalent Command: Layout-Styles-[Edit]

Syntax

```
StylesEdit(
        Name:string;
        Description:string;
        StyleType:{
                Open!=0;
                Paired!=1
        };
        EnterType:{
                HardReturn!=0;
                Off!=1;
                OffOn!=2
        };
        Level:number;
        EditCodes:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Name: Specifies the name of the style to edit. Enclose the string in quotes.

Description: Specifies the description of the style. Enclose the string in quotes.

StyleType: Specifies the type of style (open or paired).

EnterType: Specifies the effect of pressing **[Enter]** when the style is on.

Level: Specifies the outline level for the style (outline styles only). Valid values are 1 through 8.

EditCodes: Specifies whether the Styles editor is opened. If **Yes!**, subsequent formatting commands apply to the style definition. The macro should close the Styles editor with a Close command.

Omit parameters when no change is desired.

StylesProperties

Function: Modifies the properties specified in the style definition, but does not edit the style codes.

Equivalent Command: Layout-Styles-[Edit]-[Properties]

Syntax

```
StylesProperties(
        Name:string;
        Description:string;
        StyleType:{
                Open!=0;
                Paired!=1
        };
        EnterType:{
                HardReturn!=0;
                Off!=1;
                OffOn!=2
        }
)
```

Parameters

Name: Specifies the name of the style to change. Enclose the string in quotes.

Description: Specifies the description of the style. Enclose the string in quotes.

StyleType: Specifies the type of style (open or paired).

EnterType: Specifies the effect of pressing **[Enter]** when the style is on.

Omit parameters when no change is desired. Use this command only inside the Style editor.

StylesPropertiesDlg

Function: Calls the Style Properties dialog box.

Equivalent Command: Layout-Styles-[Edit]-[Properties]

Syntax
StylesPropertiesDlg()

Parameters
None

Use this command only inside the Styles Editor.

StylesRetrieve

Function: Retrieves a style file.

Equivalent Command: Layout-Styles-[Retrieve]

Syntax
StylesRetrieve(
Filename:string
)

Parameters
Filename: Specifies the name of the style file to retrieve. Enclose filename in quotes (include path if necessary).

StylesSave

Function: Saves the current styles in the specified style file.

Equivalent Command: Layout-Styles-[Save As]

Syntax
StylesSave(
Filename:string
)

Parameters
Filename: Specifies the name of the style file to save. Note: enclose filename in quotes (include path if necessary).

Tab

Function: Inserts a tab.

Equivalent Command: **[Tab]**

Syntax
Tab()

Parameters
None

TabInsert

Function: Inserts a tab at the insertion point (used mostly to insert a tab in a cell).

Equivalent Command: **[Ctrl+Tab]**

Syntax
TabInsert()

Parameters
None

TableAppendRow

Function: Adds a new row at the insertion point in the current table.

Equivalent Command: **[Alt+Shift+Insert]**

Syntax
TableAppendRow()

Parameters
None

TableBoxCaptionEdit

Function: Opens the Caption Editor for the currently selected table box.

Equivalent Command: Graphics-Table Box--Caption

Syntax
TableBoxCaptionEdit()

Parameters
None

TableBoxCaptionEdit is a non-recordable token. WPWin will record the BoxEditCaption command instead.

TableBoxCreate

Function: Creates a new table box and opens the Text Box Editor.

Equivalent Command: Graphics-Table Box-Create

Syntax
TableBoxCreate()

Parameters
None

TableBoxCreateDlg

Function: Calls the Select Editor dialog box.

Equivalent Command: Graphics-Table Box-Create

Syntax
TableBoxCreateDlg()

Parameters
None

TableBoxEdit

Function: Opens the appropriate editor (Figure, Text, or Equation) for the currently selected table box.

Equivalent Command: Graphics-Table Box-Edit

Syntax
TableBoxEdit()

Parameters
None

TableBoxEdit is a non-recordable token. WPWin will record the BoxEditor command instead.

TableBoxNewNumberDlg

Function: Calls the Table Box Number dialog box.

Equivalent Command: Graphics-Table Box-New Number

Syntax
TableBoxNewNumberDlg()

Parameters
None

TableBoxOptionsDlg

Function: Calls the Table Box Options dialog box.

Equivalent Command: Graphics-Table Box-Options

Syntax
TableBoxOptionsDlg()

Parameters
None

TableBoxPositionDlg

Function: Calls the Box Position and Size dialog.

Equivalent Command: Graphics-Table Box-Position

Syntax
TableBoxPositionDlg()

Parameters
None

TableCalculate

Function: Calculates formulas in the current table.

Equivalent Command: Layout-Tables-Calculate

Syntax
TableCalculate()

Parameters
None

The insertion point must be inside the table or this command has no effect.

TableCell

Function: Formats the current cell (the cell containing the insertion point). Multiple cells can be formatted if they are selected first.

Equivalent Command: Layout-Tables-Cell

Syntax

```
TableCell(
        Bold:{
                Off!=0;
                On!=1
        };
        Underline:{
                Off!=0;
                On!=1
        };
        DblUnderline:{
                Off!=0;
                On!=1
        };
        Italic:{
                Off!=0;
                On!=1
        };
        Outline:{
                Off!=0;
                On!=1
        };
        Shadow:{
                Off!=0;
                On!=1
        };
        SmallCap:{
                Off!=0;
                On!=1
        };
        Redline:{
                Off!=0;
                On!=1
        };
        Strikeout:{
                Off!=0;
                On!=1
        };
        Superscript:{
                Off!=0;
                On!=1
        };
        Subscript:{
                Off!=0;
                On!=1
        };
```

```
	Fine:{
		Off!=0;
		On!=1
	};
	Small:{
		Off!=0;
		On!=1
	};
	Large:{
		Off!=0;
		On!=1
	};
	VeryLarge:{
		Off!=0;
		On!=1
	};
	ExtraLarge:{
		Off!=0;
		On!=1
	};
	ColumnAppSize:{
		Off!=0;
		On!=1
	};
	Justify:{
		Left!=0;
		Full!=1;
		Center!=2;
		Right!=3;
		DecimalAlign!=4
	};
	ColumnJustify{
		Off!=0;
		On!=1
	};
	NoCalc:{
		Off!=0;
		On!=1
	};
	VertAlign:{
		Top!=0;
		Bottom!=1;
		Center!=2
	};
	Lock:{
		Off!=0;
		On!=1
	};
	Shade:{
		Off!=0;
		On!=1
	}
)
```

Parameters
FontAttributes: The font attribute parameters (Bold, Underline, etc.) specify whether a given font

attribute is on or off.

ColumnAppSize: Specifies if column size and appearance, as set in the Format Column dialog box (see TableColumn command), applies to the cell.

Justify: Specifies justification of cell contents.

ColumnJustify: Specifies justification for current column.

NoCalc: Specifies whether cell is skipped during calculations.

VertAlign: Specifies the vertical orientation for text in the cell.

Lock: Specifies whether cell is locked (contents cannot be edited).

Shade: Specifies shading for cell.

TableCellDlg

Function: Calls the Format Cell dialog box.

Equivalent Command: Layout-Table-Cell

Syntax
TableCellDlg()

Parameters
None

TableColumn

Function: Sets the formatting of cells for the current column (containing the insertion point) in a table.

Equivalent Command: Layout-Tables-Column

Syntax

```
TableColumn(
	Bold:{
		Off!=0;
		On!=1
	};
	Underline:{
		Off!=0;
		On!=1
	};
	DblUnderline:{
		Off!=0;
		On!=1
	};
	Italic:{
		Off!=0;
		On!=1
	};
	Outline:{
		Off!=0;
		On!=1
	};
	Shadow:{
		Off!=0;
		On!=1
	};
	SmallCap:{
		Off!=0;
		On!=1
	};
	Redline:{
		Off!=0;
		On!=1
	};
	Strikeout:{
		Off!=0;
		On!=1
	};
	Superscript:{
		Off!=0;
		On!=1
	};
	Subscript:{
		Off!=0;
		On!=1
	};
	Fine:{
```

```
		Off!=0;
		On!=1
	};
	Small:{
		Off!=0;
		On!=1
	};	 Large:{
		Off!=0;
		On!=1
	};
	VeryLarge:{
		Off!=0;
		On!=1
	};
	ExtraLarge:{
		Off!=0;
		On!=1
	};
	Digits:number;
	Justify:{
		Left!=0;
		Full!=1;
		Center!=2;
		Right!=3;
		DecimalAlign!=4
	};
	Width:wpunits
)
```

Parameters

FontAttributes: The font attribute parameters (Bold, Underline, etc.) specify whether a given font attribute is on or off.

Digits: Specifies the number of digits displayed for decimal numbers.

Justify: Specifies justification of cell contents.

Width: Specifies the width of the column. Value: any valid wpunits measurement.

TableColumnDlg

Function: Calls the Format Column dialog box.

Equivalent Command: Layout-Tables-Column

Syntax
TableColumnDlg()

Parameters
None

TableConvert

Function: Converts selected columns (parallel or newspaper) to table format.

Equivalent Command: Layout-Tables-Create

Syntax

```
TableConvert(
        ConvertFrom:{
                TabularColumns!=0;
                ParallelColumns!=1
        }
 )
```

Parameters

ConvertFrom: Specifies the column type of the original text.

Text must be selected before using command.

TableConvertDlg

Function: Calls the Convert Table dialog box.

Equivalent Command: Layout-Tables-Create

Syntax
TableConvertDlg()

Parameters
None

TableCreate

Function: Creates a table of the specified number or columns and rows.

Equivalent Command: Layout-Table-Create

Syntax

```
TableCreate(
        Columns:number;
        Rows:number
)
```

Parameters

Columns: Specifies the number of columns for the table. Valid values are 1 through 32.

Rows: Specifies the number of rows for the table. Valid values are 1 through 32765.

TableCreateNewDlg

Function: Calls the Create Table dialog box.

Equivalent Command: Layout-Table-Create

Syntax
TableCreateNewDlg()

Parameters
None

TableDeleteColumn

Function: Deletes one or more columns at the insertion point.

Equivalent Command: Layout-Tables-Delete-(Columns)

Syntax
TableDeleteColumn(
Count:number
)

Parameters
Count: Specifies the number of columns to delete at the insertion point (the column with the insertion point is the first column; additional columns are deleted to the right).

Only the current column is deleted if the Count: parameter is omitted.

TableDeleteDlg

Function: Calls the Delete Rows/Columns dialog box.

Equivalent Command: Layout-Tables-Delete

Syntax
TableDeleteDlg()

Parameters
None

TableDeleteRow

Function: Deletes one or more rows at the insertion point.

Equivalent Command: Layout-Tables-Delete-(Rows)

Syntax
TableDeleteRow(
Count:number
)

Parameters
Count: Specifies the number of rows to delete at the insertion point (the row with the insertion point is the first row; additional rows are deleted below it).

Only the current row is deleted if the Count: parameter is omitted.

TableEditColumn

Function: Sets the left edge and column widths for the current table.

Equivalent Command: Macro only.

Syntax

```
TableEditColumn(
        LeftEdge:wpunits;
        Width1:wpunits;
        ...
        Width32:wpunits
)
```

Parameters

LeftEdge: Specifies the distance from the left margin to the left edge of the table.

Width**x**: Specifies the width of the column, indicated by x, a number from 1 to 32.

TableFormula

Function: Inserts the specified formula into the current cell, and optionally copies the formula down rows or across columns.
Equivalent Command: Layout-Tables-Formula

Syntax

```
TableFormula(
        Formula:string;
        Copy:{
                To!=0;
                Down!=1;
                Right!=2
        };
        Cell:string;
        Count:number
)
```

Parameters

Formula: Specifies the formula to use. Enclose the string in quotes.

Copy: Specifies that the formula is copied to other cells.

Cell: If Copy: is set to **To!**, specifies the cell coordinates to copy the formula to. Enclose the string in quotes. Omit if Copy: is set to **Down!** or **Right!**.

Count: If Copy: is set to **Down!** or **Right!**, specifies the number of rows **(Down!**) or columns (**Right!**) to receive the copied formula. Omit if Copy: is set to **To!**.

TableFormulaDlg

Function: Calls the Tables Formula dialog box.

Equivalent Command: Layout-Tables-Formula

Syntax
TableFormulaDlg()

Parameters
None

TableInsertColumn

Function: Inserts one or more columns to the right of the insertion point.

Equivalent Command: Layout-Tables-Insert-(Columns)

Syntax
TableInsertColumn(
Count:number
)

Parameters
Count: Specifies the number of inserted columns (total number of columns cannot exceed 32).

TableInsertDlg

Function: Calls the Insert Rows/Columns dialog box.

Equivalent Command: Layout-Tables-Insert

Syntax
TableInsertDlg()

Parameters
None

TableInsertRow

Function: Inserts one or more rows below the insertion point.

Equivalent Command: Layout-Tables-Insert-(Rows)

Syntax
TableInsertRow(
	Count:number
)

Parameters
Count: Specifies the number of inserted rows (total number of rows cannot exceed 32768).

TableJoin

Function: Joins selected cells into a single cell. This command is ignored if two or more cells are not selected.

Equivalent Command: Layout-Tables-Join

Syntax
TableJoin()

Parameters
None

TableLineDlg

Function: Calls the Table Lines dialog box.

Equivalent Command: Layout-Tables-Lines

Syntax
TableLineDlg()

Parameters
None

TableLines

Function: Specifies the format of lines for the current cell (or all currently selected cells).

Equivalent Command: Layout-Table-Lines

Syntax

```
TableLines(
        Left:{
                None!=0;
                Single!=1;
                Double!=2;
                Dashed!=3;
                Dotted!=4;
                Thick!=5;
                ExtraThick!=6
        };
        Right:{
                (same as Left)
        };
        Top:{
                (same as Left)
        };
        Bottom:{
                (same as Left)
        };
        Inside:{
                (same as Left)
        };
        Outside:{
                (same as Left)
        }
)
```

Parameters

The parameters specify the line format (none, single, double, dashed, dotted, thick, or extra thick) for the left, right, top, bottom, inside, and outside lines of table cells.

TableOptions

Function: Sets table options.

Equivalent Command: Layout-Tables-Options

Syntax

```
TableOptions(
        Columns:number;
        Rows:number;
        LeftMargin:wpunits;
        RightMargin:wpunits;
        TopMargin:wpunits;
        BottomMargin:wpunits;
        TablePosition:{
                Left!=0;
                Right!=1;
                Center!=2;
                Full!=3;
                FromLeftEdge!=4
        };
        FromLeftEdge:wpunits;
        HeaderRows:number;
        ShadePercent:number;
        NegDisplay:{
                Minus!=0;
                Paren!=1
        };
        DisableLocks:{
                Off!=0;
                On!=1
        }
)
```

Parameters

Columns: Specifies the number of columns (from 1 to 32) for the current table.

Rows: Specifies the number or rows (from 1 to 32765) for the current table.

LeftMargin: Specifies the amount of inside space between the left margin of the table and the table contents. Value: any valid wpunits measurement.

RightMargin: Specifies the amount of inside space between the right margin of the table and the table contents. Value: any valid wpunits measurement.

TopMargin: Specifies the amount of inside space between the top margin of the table and the table contents. Value: any valid wpunits measurement.

BottomMargin: Specifies the amount of inside space between the bottom margin of the table and the table contents. Value: any valid wpunits measurement.

TablePosition: Specifies the position of the table on the page.

FromLeftEdge: If TablePosition: is set to **FromLeftEdge!**, specifies the distance between the left edge of the page and the left edge of the table. Omit this parameter if TablePosition: is set to another value.

HeaderRows: Specifies the number of rows of the table that are header rows (these rows repeat for each

page if the table spans more than one page). Omit this parameter, or set it to 0, if you don't want header rows.

ShadePercent: Specifies the default shading of all cells in the table. Valid values are 0 (no shading) to 100 (black).

NegDisplay: Specifies how negative numbers are displayed.

DisableLocks: Specifies whether cells in the table can be locked.

TableOptionsDlg

Function: Calls the Table Options dialog box.

Equivalent Command: Layout-Tables-Options

Syntax
TableOptionsDlg()

Parameters
None

TableRow

Function: Specifies height for the current cell (or group of cells if more than one cell is selected).

Equivalent Command: Layout-Tables-Row

Syntax

```
TableRow(
        SingleLine:{
                Off!=0;
                On!=1
        };
        FixedHeight:{
                Off!=0;
                On!=1
        };
        Height:wpunits
)
```

Parameters

SingleLine: Specifies whether cells are be confined to single lines.

FixedHeight: Specifies whether cells are set to a fixed height.

Height: If FixedHeight: is set to **On!**, specifies the height of the cells. Value: any valid wpunits measurement.

TableRowDlg

Function: Calls the Format Row dialog box.

Equivalent Command: Layout-Tables-Row

Syntax
TableRowDlg()

Parameters
None

TableSplit

Function: Splits the selected cell (or group of selected cells) into two or more rows or columns.

Equivalent Command: Layout-Tables-Split

Syntax

```
TableSplit(
        Split:{
                Rows!=0;
                Columns!=1
        };
        Count:integer
)
```

Parameters

Split: Specifies whether the cell is to be split into two or more rows or columns.

Count: Specifies the number of rows or columns to create in the split. The total number of columns for the table cannot exceed 32; the total number of rows for the table cannot exceed 32765.

TableSplitDlg

Function: Calls the Split Column/Row dialog box.

Equivalent Command: Layout-Table-Split

Syntax
TableSplitDlg()

Parameters
None

TabSet

Function: Sets one or more tabs.

Equivalent Command: Layout-Line-Tab Set

Syntax

```
TabSet(
        Origin:{
                LeftEdge!=0;
                LeftMargin!=1
        };
        Type1:{
                Left!=0;
                Center!=1;
                Right!=2;
                Decimal!=3;
                LeftDot!=4;
                CenterDot!=5;
                RightDot!=6;
                DecimalDot!=7
        };
        Pos1:wpunits;
        ...
        Type40:{
                Left!=0;
                Center!=1;
                Right!=2;
                Decimal!=3;
                LeftDot!=4;
                CenterDot!=5;
                RightDot!=6;
                DecimalDot!=7
        };
        Pos40:wpunits
)
```

Parameters

Origin: Specifies the origin of the tab, either left edge (absolute) or left margin (relative).

Type**x**: Specifies the type of tab, where x identifies the tab and is a number from 1 to 40.

Pos**x**: Specifies the position of the tab stop, where x identifies the tab and is a number from 1 to 40.
Value: any valid wpunits measurement.

TabSetDlg

Function: Calls the TabSet dialog box.

Equivalent Command: Layout-Line-Tab Set

Syntax
TabSetDlg()

Parameters
None

TextBoxCaptionEdit

Function: Opens the Caption Editor for the currently selected text box.

Equivalent Command: Graphics-Text Box--Caption

Syntax
TextBoxCaptionEdit()

Parameters
None

TextBoxCaptionEdit is a non-recordable token. WPWin will record the BoxEditCaption command instead.

TextBoxCreate

Function: Creates a new text box and opens the Text Box Editor

Equivalent Command: Graphics-Text Box-Create

Syntax
TextBoxCreate()

Parameters
None

TextBoxCreateDlg

Function: Calls the Select Editor dialog box.

Equivalent Command: Graphics-Text Box-Create

Syntax
TextBoxCreateDlg()

Parameters
None

TextBoxEdit

Function: Opens the appropriate editor (Figure, Text, or Equation) for the currently selected text box.

Equivalent Command: Graphics-Text Box-Edit

Syntax
TextBoxEdit()

Parameters
None

TextBoxEdit is a non-recordable token. WPWin will record the BoxEditor command instead.

TextBoxNewNumberDlg

Function: Calls the Text Box Number dialog box.

Equivalent Command: Graphics-Text Box-New Number

Syntax
TextBoxNewNumberDlg()

Parameters
None

TextBoxOptionsDlg

Function: Calls the Text Box Options dialog box.

Equivalent Command: Graphics-Text Box-Options

Syntax
TextBoxOptionsDlg()

Parameters
None

TextBoxPositionDlg

Function: Calls the Box Position and Size dialog.

Equivalent Command: Graphics-Text Box-Position

Syntax
TextBoxPositionDlg()

Parameters
None

TextBoxRotate

Function: Rotates text in the currently open text box editor (this command should only be used in the text box editor).

Equivalent Command: Graphics-Text Box-Edit-[Rotate]

Syntax

```
TextBoxRotate(
        RotationValue:{
                Rotate0!=0;
                Rotate90!=1;
                Rotate180!=2;
                Rotate270!=3
        }
)
```

Parameters

RotationValue: Specifies the rotation (in degrees clockwise) of the text.

TextBoxRotateDlg

Function: Calls the Rotate Text dialog box.

Equivalent Command: Graphics-Text Box-Edit-[Rotate]

Syntax
TextBoxRotateDlg()

Parameters
None

Thesaurus

Function: Calls the WPWin Thesaurus utility.

Equivalent Command: Tools-Thesaurus

Syntax
Thesaurus()

Parameters
None

ToADefine

Function: Defines a table of authority list. Places a Table of Authorities code marker in the document; when the document is generated the table of authorities is placed here.

Equivalent Command: Tools-Table of Authorities-Define

Syntax

```
ToADefine(
        SectionNumber:number;
        BlankLineBetween:{
                No!=0;
                Yes!=1
        };
        DotLeaders:{
                No!=0;
                Yes!=1
        };
        AllowUnderline:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

SectionNumber: Specifies the section number. Value should be between 1 and 16.

BlankLineBetween: Specifies whether blank lines are to be inserted between table of authorities entries.

DotLeaders: Specifies whether dot leaders are automatically included between the entry and the page number reference.

AllowUnderline: Specifies whether underlining is allowed within the table of authorities list.

ToADefineDlg

Function: Calls the Define Table of Authorities dialog box.

Equivalent Command: Tools-Table of Authorities-Define

Syntax
ToADefineDlg()

Parameters
None

ToAEditFull

Function: Opens an editing screen containing the preceding table of authorities full form mark. Subsequent macro commands affect the newly opened editing window. Use the Close command to return to the main WPWin document window.

Equivalent Command: Tools-Mark Text-ToA Edit Full Form

Syntax
ToAEditFull()

Parameters
None

ToAMarkFull

Function: Marks the selected text for a table of authorities entry. If no text is selected, this command is ignored.

Equivalent Command: Tools-Mark Text-ToA Full Form

Syntax
ToAMarkFull()

Parameters
None

ToAMarkShort

Function: Sets a mark at the insertion point for a specified short form in a table of authorities.

Equivalent Command: Tools-Mark Text-ToA Short Form

Syntax
ToAMarkShort(
    ShortFormText:string
)

Parameters
ShortFormText: Specifies the text to use for the short form. Enclose the string in quotes.

ToAMarkShortDlg

Function: Calls the Mark ToA Short Form dialog box.

Equivalent Command: Tools-Mark Text-ToA Short Form

Syntax
ToAMarkShortDlg()

Parameters
None

ToCDefine

Function: Defines a table of contents list. Places a Table of Contents code marker in the document; when the document is generated the table of contents is placed here.

Equivalent Command: Tools-Table of Contents-Define

Syntax

```
ToCDefine(
        NumLevels:number;
        WrapLastLvl:{
                No!=0;
                Yes!=1
        };
        Level1Style:{
                NoNumbering!=0;
                NumFollowsEntry!=1;
                FollowsInParens!=2;
                FlushRight!=3;
                FlushLeaders!=4
        };
        ...
        Level5Style:{
                NoNumbering!=0;
                NumFollowsEntry!=1;
                FollowsInParens!=2;
                FlushRight!=3;
                FlushLeaders!=4         }
)
```

Parameters

NumLevels: Specifies the number of levels to use in the table of contents. Valid values are 1 to 5.

WrapLastLvl: Specifies whether the final level is wrapped.

Level**x**Style: Specifies the format for the table of contents level, where x is the level (1 to 5).

ToCDefineDlg

Function: Calls the Define Table of Contents dialog box.

Equivalent Command: Tools-Table of Contents-Define

Syntax
ToCDefineDlg()

Parameters
None

ToCMark

Function: Marks the selected text for a table of contents entry. If no text is selected, this command is ignored.

Equivalent Command: Tools-Mark Text-Table of Contents

Syntax

```
ToCMark(
        LevelNumber:{
                Level1!=0;
                Level2!=1;
                Level3!=2;
                Level4!=3;
                Level5!=4
        }
)
```

Parameters

LevelNumber: Specifies the level number (1 to 5) for the table of contents entry.

ToCMarkDlg

Function: Calls the Mark Table of Contents dialog box.

Equivalent Command: Tools-Mark Text-Table of Contents

Syntax
ToCMarkDlg()

Parameters
None

Type

Function: Inserts text at the insertion point.

Equivalent Command: Macro only.

Syntax
Type(
Text:string
)

Parameters
Text: Specifies the text to insert. Enclose the string in quotes.

TypeChar

Function: Inserts a specific WP character at the insertion point. TypeChar converts a character value to a printable WP character.

Equivalent Command: Macro only.

Syntax

```
TypeChar(
        Character:number;
        CharacterSet:number;
        CharacterOffset:number
)
```

Parameters

Character: Specifies the character value. The value is computed using the following formula:

(Character Set Number * 256) + Character Number

Omit Character: when using CharacterSet: and CharacterOffset:.

- CharacterSet: Specifies the character set value to use.
- CharacterOffset: Specifies the character number defined in the character set you want to use.

TypesetBaseline

Function: Determines whether first text baseline is placed even with the top margin of the page.

Equivalent Command: Tools-Typesetting-
(First Baseline at Top Margin)

Syntax

```
TypesetBaseline(
        State:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

State: Specifies whether baseline alignment is turned on or off.

TypesetDlg

Function: Calls the Typesetting dialog box.

Equivalent Command: Tools-Typesetting

Syntax
TypesetDlg()

Parameters
None

TypesetJustifyLimits

Function: Sets minimum and maximum space between words in justified text.

Equivalent Command: Layout-Typesetting

Syntax
TypesetJustifyLimits(
CompressedTo:number;
ExpandedTo:number)

Parameters
CompressedTo: Specifies the minimum space between words. Expressed as a percentage (whole number only) between 0 and 100.

ExpandedTo: Specifies the maximum space between words. Expressed as a percentage (whole number only) between 100 and 1000.

TypesetKerning

Function: Specifies whether WPWin should automatically kern letter pairs.

Equivalent Command: Layout-Typesetting

Syntax

```
TypesetKerning(
        Kerning:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Kerning: Specifies whether automatic kerning is turned on or off.

TypesetLeadingAdjust

Function:Specifies leading between lines (spaces between hard return and soft return lines are controlled separately).

Equivalent Command: Layout-Typesetting

Syntax

```
TypesetLeadingAdjust(
        BetweenLines:wpunits;
        BetweenParagraphs:wpunits
)
```

Parameters

BetweenLines: Specifies leading between soft return lines. Value: any valid wpunits measurement.

BetweenParagraphs: Specifies leading between hard return lines (paragraphs). Value: any valid wpunits measurement.

TypesetLetterspace

Function: Determines the spacing between characters on a line.

Equivalent Command: Layout-Typesetting

Syntax

```
TypesetLetterspace(
        Mode:{
                Normal!=0;
                WPOptimal!=1;
                PercentOfOptimal!=2
        };
        Percent:integer
)
```

Parameters

Mode: Specifies letterspace mode: Normal (set by font); WPOptimal (sets width to one-third font width); PercentOfOptimal (set manual spacing).

Percent: When Mode: is set to **PercentOfOptimal!** .

TypesetManualKerningDlg

Function: Calls the Manual Kerning dialog box.

Equivalent Command: Layout-Typesetting-Manual Kerning

Syntax
TypesetManualKerningDlg()

Parameters
None

TypesetUnderlineOptions

Function: Sets underline options.

Equivalent Command: Layout-Typesetting

Syntax

```
TypesetUnderlineOptions(
        Spaces:{
                No!=0;
                Yes!=1
        };
        Tabs:{
                No!=0;
                Yes!=1
        }
)
```

Parameters

Spaces: Specifies whether spaces between words are underlined.

Tabs: Specifies whether white space produced by tabs is underlined.

TypesetWordSpace

Function: Specifies the spacing between words

Equivalent Command: Layout-Typesetting

Syntax

```
TypesetWordSpace(
        Mode:{
                Normal!=0;
                WPOptimal!=1;
                PercentOfOptimal!=2
        };
        Percent:integer
)
```

Parameters

Mode: Specifies wordspace mode: Normal (set by font); WPOptimal (sets width to one-third font width); PercentOfOptimal (set manual spacing).

Percent: When Mode: is set to **PercentOfOptimal!**.

Undelete

Function: Retrieves previously deleted text from one of three undelete buffers.

Equivalent Command: Edit-Undelete

Syntax

```
Undelete(
        WhichBuffer:{
                BufferOne!=1;
                BufferTwo!=2;
                BufferThree!=3
        }
)
```

Parameters

WhichBuffer: Specifies the buffer that contains the text to undelete.

UndeleteDlg

Function: Calls the Undelete dialog box.

Equivalent Command: Edit-Undelete

Syntax
UndeleteDlg()

Parameters
None

Undo

Function: Reverses the last edit change

Equivalent Command: Edit-Undo

Syntax
Undo()

Parameters
None

UserBoxCaptionEdit

Function: Opens the Caption Editor for the currently selected user box.

Equivalent Command: Graphics-User Box--Caption

Syntax
UserBoxCaptionEdit()

Parameters
None

UserBoxCaptionEdit is a non-recordable token. WPWin will record the BoxEditCaption command instead.

UserBoxCreate

Function: Creates a new table box and opens the Text Box Editor

Equivalent Command: Graphics-User Box-Create

Syntax
UserBoxCreate()

Parameters
None

UserBoxCreateDlg

Function: Calls the Select Editor dialog box.

Equivalent Command: Graphics-User Box-Create

Syntax
UserBoxCreateDlg()

Parameters
None

UserBoxEdit

Function: Opens the appropriate editor (Figure, Text, or Equation) for the currently selected user box.

Equivalent Command: Graphics-User Box-Edit

Syntax
UserBoxEdit()

Parameters
None

UserBoxEdit is a non-recordable token. WPWin will record the BoxEditor command instead.

UserBoxNewNumberDlg

Function: Calls the User Box Number dialog box.

Equivalent Command: Graphics-User Box-New Number

Syntax
UserBoxNewNumberDlg()

Parameters
None

UserBoxOptionsDlg

Function: Calls the User Box Options dialog box.

Equivalent Command: Graphics-User Box-Options

Syntax
UserBoxOptionsDlg()

Parameters
None

UserBoxPositionDlg

Function: Calls the Box Position and Size dialog.

Equivalent Command: Graphics-User Box-Position

Syntax
UserBoxPositionDlg()

Parameters
None

UserFunction

Function: Executes a third-party program (typically a dynamic link library, or DLL) previous "registered" with WPWin.

Equivalent Command: Macro only.

Syntax
UserFunction(
Action:string
)

Parameters
Action: Specifies the character string to "pass" to the third- party program. This string consists of two parts: Signature:Action

- The Signature is a four character string that uniquely identifies the third-party application.
- The Action is any string recognized by the third-party program.

Separate the Signature and Action parts with a colon:

Example: UserFunction("WPTP:MacroCommands")

VertLineCreateDlg

Function: Calls the Create Vertical Line dialog box.

Equivalent Command: Graphics-Line-Vertical

Syntax
VertLineCreateDlg()

Parameters
None

VertLineEditDlg

Function: Calls the Edit Vertical Line dialog box.

Equivalent Command: Graphics-Line-Vertical

Syntax
VertLineEditDlg()

Parameters
None

WindowCascade

Function: Arranges document windows in cascade style.

Equivalent Command: Windows-Cascade

Syntax
WindowCascade()

Parameters
None

WindowTile

Function: Arranges document windows in tile style.

Equivalent Command: Windows-Tile

Syntax
WindowTile()

Parameters
None

WordCountDlg

Function: Calls the Word Count dialog box.

Equivalent Command: Tools-Word Count

Syntax
WordCountDlg()

Parameters
None

WPCharactersDlg

Function: Calls the WP Characters dialog.

Equivalent Command: Font-WP Characters

Syntax
WPCharactersDlg()

Parameters
None

Zoom 

Function: Zooms the display by the specified amount (between 50 and 400 percent).

Equivalent Command: Macro only

Syntax
Zoom (
Percent: number)
)

Parameters
Percent: Integer (whole number) value between 50 and 400.

Zoom100

Function: Zooms the display to 100 percent.

Equivalent Command: View-Zoom-(100%)

Syntax
Zoom100()

Parameters
None

Zoom150 

Function: Zooms the display to 150 percent.

Equivalent Command: View-Zoom-(150%)

Syntax
Zoom150()

Parameters
None

Zoom200 

Function: Zooms the display to 200 percent.

Equivalent Command: View-Zoom-(200%)

Syntax
Zoom200()

Parameters
None

Zoom50 

Function: Zooms the display to 50 percent.

Equivalent Command: View-Zoom-(50%)

Syntax
Zoom50()

Parameters
None

Zoom75 

Function: Zooms the display to 75 percent.

Equivalent Command: View-Zoom-(75%)

Syntax
Zoom75()

Parameters
None

ZoomDlg 

Function: Calls the Zoom dialog.

Equivalent Command: View-Zoom

Syntax
ZoomDlg()

Parameters
None

ZoomToPageWidth 

Function: Zooms the display to full page width.

Equivalent Command: View-Zoom-(To Page Width)

Syntax
ZoomToPageWidth()

Parameters
None

Macro Command Types

There are 11 categories of macro programming commands. With only a couple of exceptions, these do not include product function commands such as **AdvanceDlg**:

- User interface
- Flow control
- Macro and subroutine termination
- External conditioning handling
- Macro execution
- Macro execution control
- Variables
- Programming aids
- Dynamic link library
- Dialog box
- Special purpose

Note that some commands serve double duty. For example, the **Return** command is found in both the Macro and Subroutine Termination commands and Flow Control commands.

User Interface Commands

The user interface commands allow input from the user and/or display a message on the screen. The user interface commands are:

Beep -- Sounds a short warning tone.

EndPrompt -- Ends a Prompt message box.

GetNumber -- Displays a message box for number entry.

GetString -- Displays a message box for text entry.

GetUnits -- Displays a message box for WordPerfect for measurement units entry.

MacroStatusPrompt -- Displays a message in the status line at the bottom of the WordPerfect window.
(Note: **MacroStatusPrompt** is a product function, not a programming command).

Prompt -- Displays a message box.

Flow Control Commands

Flow control commands redirect macro execution depending on external criteria (such as user input). The flow control commands are:

AssertCancel -- Generates a "cancel condition."

AssertError -- Generates an "error condition."

AssertNotFound -- Generates a "not found condition."

Call -- Calls a subroutine.

Case -- Establishes one or more branches.

Case Call -- Establishes one or more branches, with each branch executed as a subroutine.

Chain -- Executes another macro.

Else -- Establishes a FALSE condition for an If statement.

EndFor -- Ends a For and ForEach loop.

EndIf -- End an If statement.

EndWhile -- Ends a While loop.

For -- Establishes a self-contained "counter" loop.

ForEach -- Applies one or more commands or routines to a single variable.

Go -- Jumps to a labeled routine.

If -- Establishes an IF conditional statement and what happens when the statement is TRUE.

Label -- Names a routine.

OnCancel -- Provides an alternative response in case the macro is canceled.

OnError -- Provides an alternative response in case a error occurs during macro execution.

OnNotFound -- Provides an alternative response in case a search fails during macro execution.

Quit -- Stops all macro execution.

Repeat -- Repeats a loop until a condition is met.

Run -- Temporarily executes another macro, then returns to the original macro.

Return -- Returns from a nested macro (using Run) or returns from a subroutine.

ReturnCancel -- Establishes the default condition for the OnCancel command.

ReturnError -- Establishes the default condition for the OnError command.

ReturnNotFound -- Establishes the default condition for the OnNotFound command.

Until -- Establishes a condition for the Repeat command.

While -- Repeats a loop while a condition is met.

Macro and Subroutine Termination Commands

The macro and subroutine termination commands end macro execution, or break out of subroutine and continue with the rest of the macro. The macro and subroutine termination commands are:

AssertCancel -- Generates a "cancel condition."

AssertError -- Generates an "error condition."

AssertNotFound -- Generates a "not found condition."

Quit -- Stops the macro.

Return -- Returns from a nested macro (using Run) or returns from a subroutine.

External Condition Handling Commands

The external condition handling commands specify how a condition that occurs outside the macro is manipulated. In the case of the ASSERT commands, they generate or create the condition. The external condition handling commands are:

AssertCancel -- Generates a "cancel condition."

AssertError -- Generates an "error condition."

AssertNotFound -- Generates a "not found condition."

CancelOff -- Disables the Cancel key (Esc) from terminating the macro.

CancelOn -- Enables the Cancel key.

OnCancel -- Provides an alternative response in case the macro is canceled.

OnError -- Provides an alternative response in case a error occurs during macro execution.

OnNotFound -- Provides an alternative response in case a search fails during macro execution.

Macro Execution Commands

The Macro execution commands run a macro.

Chain -- Starts a new macro.

Run -- Temporarily executes another macro, then returns to the original macro.

Macro Execution Control Commands

Macro execution commands effect the operation of macros by controlling either the display or macro execution speed, or by delay until the next command is executed. The macro execution control commands are:

Display (Off!) -- Macro does not display text as it is entered by macro. (Note: Display is a product function, not a programming command.)

Display (On!) -- Macro displays text as it is entered by macro. (Note: Display is a product function, not a programming command.)

Speed -- Slows sown macro execution.

Wait -- Waits a predetermined time before continuing.

Variable Commands

The variable commands assign values to variables or determine some characteristic or state of a variable. The variable commands are:

Assign -- Assigns a value to a variable.

ByteLen -- Counts the number of bytes in a text string.

BytePos -- Checks for the presence of a particular character in a text string, and assigns its location (in bytes) in text to a variable.

GetWPData -- Retrieves the value of any of a number of internal system variables.

GetNumber -- Displays a message box and assigns the entered number to a variable.

GetString -- Displays a message box and assigns the entered text to a variable.

GetUnits -- Displays a message box and assigns the entered WordPerfect measurement units to a variable.

Fraction -- Assigns just the fractional part of a floating-point number (such "45" in 123.45) to a variable.

Integer -- Assigns just the integer part of a floating- point number to a variable (such as "123" in 123.45) to a variable.

Menu -- Displays a pop-up menu of choices and assigns the selection to a variable.

MergeVariableGet -- Retrieves the value of a global (i.e. not {LOCAL}) merge variable.

MergeVariableSet -- Assigns a value to a global merge variable.

NumStr -- Converts a number to a text string.

StrLen -- Counts the number of characters in a text string.

StrNum -- Converts a text string to a number.

StrPos -- Checks for the presence of a particular character in a text string, and assigns its location in text to a variable.

SubByte -- Captures a substring (part of a text string) of a variable, specified in bytes.

SubStr -- Captures a substring (part of a text string) of a variable.

Programming Aid Commands

Programming air commands help make programming and "debugging" macro easier. The programming aid commands are:

// -- Provides non-executing comment with the macro.

Beep -- Sounds a short tone (audible feedback).

Speed -- Controls the playback speed of a macro.

Dynamic Link Library Commands

The following commands allow macros to access some of the Dynamic Link Library (DLL) modules that are a part of Windows and WordPerfect.

DLLCall -- Accesses a particular function within a DLL file.

DLLFree -- Releases the DLL file from use within the macro, and restores the memory previously occupied by the link to the DLL.

DLLLoad-- Locates and links to a specific DLL file.

Dialog Box Commands

The following commands allow macros to create custom dialog boxes, using over a dozen and a half types of dialog box controls. The dialog box controls available in the macro language are the same ones used in WPWin.

The Dialog box commands are not available in the initial 11/4/91 release of WPWin. They were introduced in the May 1992 interim release, and can only be used with versions of WPWin dated May 1992 or later (actual file date 4/30/92).

DialogAddCheckBox -- Adds a checkbox control for turning an option on or off. The selection of the checkbox is exclusive of the setting of other check box controls in the same dialog box or group.

DialogAddColorWheel -- Adds a color wheel for choosing a font color.

DialogAddComboBox -- -- Adds a list of selection items in any of three styles: simple, dropdown, and droplist. Selection items are placed into the combo box with the DialogAddListItem command.

DialogAddCounter -- Add a numeric counter that includes an edit box for manually entering a number, and "spin" buttons for incrementing and decrementing the value within the edit box via the mouse.

DialogAddEditBox -- Adds an editing box for text entry. Text entry can span one line, or several, and can be made to automatically word wrap at the end of each line.

DialogAddFilenameBox -- Adds a filename entry box and a directory ("browse") button. Click the browse button to select from a file list, or enter the filename into the entry box.

DialogAddFrame -- Adds a frame; used to logically group controls so that their functions are shown to be inter-related. You can select color of the frame.

DialogAddGroupBox -- Adds a group box for containing like items. Especially useful for containing radio buttons. When confinined in a group box, only one button can be selected at a time.

DialogAddHLine -- Adds a horizontal line. Used for appearance only.

DialogAddHotSpot -- Adds an "invisible" button inside the dialog box. Most often this button is placed over an icon (produced with DialogAddIcon), and behaves like a push button control.

DialogAddIcon -- Adds an icon. The icon is static; that is it serves mostly for appearance, but it can also be used with a hot spot control (with DialogAddHotSpot) to create a "picture button."

DialogAddListBox -- Adds a list box, containing selection items. Selection items are placed into the combo box with the DialogAddListItem command.

DialogAddListItem -- Adds an item to a combo, list box, or popup box.

DialogAddPopUpButton -- Adds a popup button that when pressed displays a list of selectable items. Selection items are placed into the combo box with the DialogAddListItem command.

DialogAddPushButton -- Adds a push button. The size and wording on the push button can be defined.

DialogAddRadioButton -- Adds a radio button. When placed inside a group box (using DialogAddGroupBox only one radio button can be active at a time.

DialogAddScrollBar -- Adds a vertical or horizontal scroll bar.

DialogAddText -- Adds static (non-changing) text to the dialog box. You can define the justification of the text if it spans more than one line, and specify a background box (recessed or shadow).

DialogAddViewer -- Adds a viewer control for viewing the contents of files. The viewer works the same as the one in WPWin.

DialogAddVLine -- Adds a vertical line. Used for appearance only.

DialogDefine -- Defines a dialog box, with a specified size and style. The dialog box must be defined first before controls can be added.

DialogDestroy -- Removes the dialog box and all its associated controls from memory.

DialogDisplay -- Displays the specified dialog box after the box has bbeen defined and all the controls have been added.

Special Purpose Commands

Some WordPerfect macro programming commands defy easy organization. These extend the usefulness of the macro language and are most often with other commands.

Address -- References the address of a variable, rather than the actual contents of the variable. For use with the DLL commands only.

AND, **NOT**, **OR**, **XOR** -- Logical operators for expression comparisons or bit-wise arithmetic.

Application, **Default**, **EndApp**, **File**, **NewDefault** -- Specifies the applications (such as WPWin) used in the macro.

AnsiString, **OemString**, **WPString** -- Casts a text string in a specific format. For use with the DLL commands only.

Bool, **Byte**, **Char**, **DWord**, **Real**, **String**, **Word** -- Casts a variable as a specific type. For use with the DLL commands only.

Centimeters, **DefaultUnits**, **Inches**, **Points**, **WPUnits** -- Specifies a units of measure.

Div, **Mod** -- Integer math operators.

FALSE, **TRUE** -- Specifies FALSE/TRUE (yes/no) condition.

LoWord, **HiWord** -- Extracts the lower or upper two bytes (word) in a four byte number. For use with the DLL commands only.

Logical Operators

Logical operators (they aren't commands) test for TRUE/FALSE conditions, and they also can manipulate numbers on a bit-by-bit basis. In WPWin, logical operators can be used to:

- Create compound expressions where only one or all of the elements in the expression are used in the test.
- Perform "bit-wise" math on numbers, where calculations are made by comparing the individual bits that make up a number, rather than the value of the numbers themselves.

Note that the logical operators are in all caps. This is by convention, and is by no means a requirement. I prefer the all- caps because it differentiates the logical operators from programming commands.

AND -- Used to create compound expressions, or to combine the bit values of numbers. Both parts of the **AND** expression must be TRUE for the result to be TRUE. For example: Suppose an expression reads: "If it's dark outside AND raining, then bring along a waterproof flashlight." You don't need the waterproof flashlight if it's not dark (obviously), and if it's not raining.

FALSE -- Indicates a FALSE condition.

NOT -- Reverses the meaning of an expression. That is, if an expression is TRUE, **NOT** makes it FALSE.

OR -- Used to create compound expressions, or to combine the bit values of numbers. Either, or both, parts of an **OR** expression can be TRUE for the result to be TRUE. For example: Suppose an expression reads: "If you have a cat OR a dog, then I'll need to bring along my hay fever medicine." A dog, or cat, or both cat and dog means sneezes and wheezes.

TRUE -- Indicates a TRUE condition.

XOR -- Like **OR**, but with a twist: if both parts of an **XOR** expression are true, then the result is FALSE.

Units of Measurement Commands and Values

The **Centimeter**, **Inches**, **Points**, and **WPUnits** values are used with DefaultUnits command to set a new default for units of measurement used within WPWin.

Centimeters -- Sets the default units of measurement to centimeters.

DefaultUnits -- Used with Centimeter, Inches, Points, or WPUnits to establish a new default for units of measurement. Once a default is set, you can enter values without worrying about indicating the units of measurement each time. For example, if the units of measurement is set to inches, you can enter 2.5, rather than 2.5".

Inches -- Sets the default units of measurement to inches.

Points -- Sets the default units of measurement to points. Seventy two points equals one inch.

WpUnits -- Sets the default units of measurement to WordPerfect units. One WPUnit is equal to 1/1200 of an inch.

// (Comment)

The // command inserts explanatory comments in a macro. The comment ends at the first hard return that is encountered. WPWin limits all line boundaries in a macro to 512 bytes or less. If you need to enter a comment longer than 512 bytes (typically 512 characters), add a new line break by pressing the **[Enter]** key, and type a new comment.

Example

```
// This is a simple comment

// This is another comment that spans more than one line.  Though it wraps
to the second line, the first line ends with a soft return.
```

You can place the // comment characters anywhere on the line. WPWin will treat all the text on that after the // as a comment.

Example

```
Assign (Title;"WordPerfect")  // Assigns WordPerfect to Title
```

Comments are ignored when the macro is compiled, so they do not affect the speed of macro execution. However, lots of comments increase the file size of macros, and do take longer to compile.

Application

The **Application** command identifies the WordPerfect Corp. Windows programs that are used with the macro. The command is not required if the macro does not contain any product function commands (recall that a product function command is specific to the program, whereas programming commands are generic for all WordPerfect Corp. Windows applications).

Most often, you'll use the **Application** command once at or near the beginning of the macro. In all cases, it must precede any product function commands, like **HardReturn** or **Type**. The **Application** command uses the following syntax:

Application (Product Prefix;Product ID;**Default**;Filename)

Product Prefix

The Product Prefix is the unique two-letter code that can be used to mark product function codes that belong to different applications.

Product ID

The Product ID is a unique four-letter code that specifies the application used in the macro. For WordPerfect for Windows the product ID is wpwp (capitalization doesn't matter).

Default

Default indicates that unless otherwise noted, the product functions encountered in the macro belong to the application identified in the Product ID argument. Without the **Default** command you have to append a product prefix to all product function commands.

Example

```
Application (wp;wpwp;Default)
Type ("This is a test")
     //   The wp product prefix is the default, so the Type command
          Doesn't need a prefix.

Application (wp;wpwp)
wp.Type ("This is a test")
     //   No product prefix is the default; must include the
          appropriate product prefix in front of all product
          function commands.

Application (my.wpwp)
my.Type ("This is a test")
     //   Same as above, but with a different user-defined
          product prefix.
```

Filename

The descriptions for all product function commands are contained in a Product Interface Description (PID) file. For the US version of WPWin this file is named WPWPUS.WCD (other language versions are named WPWPxx.WCD, where xx is the unique two-letter ID for the supported language). When the macro is compiled, WPWin uses the PID file to "look up" all the product function commands used in the macro.

Filename specifies a PID file to use. This argument is optional; if you don't include a filename, WPWin will use the default PID file, as entered under the [PID] section of the WPC.INI file. It's safe to leave out the Filename parameter if you know your macros will never be compiled with a copy of WPWin that uses a different language module. However, your macros may not run properly if they are compiled with a different PID file than the one you anticipate. For consistency, the US version of the WPWin PID file, WPWPUS.WCD, should be included in all language versions of WPWin.

WPWin uses the PID file only when the macro is compiled. In this way, you can distribute your macros to users without regard to PID files and differences in language. However, should the user edit or resave the macro, WPWin will insist on re-compiling it. You may want to make your macros Read-only to prevent alteration.

AssertCancel

AssertError

AssertNotFound

The **AssertCancel**, **AssertError**, and **AssertNotFound** commands generate ("assert") the associated condition during macro execution. The Assert commands are used without an argument.

Use can the commands for testing purposes, or to cause a cancel, error, or not found condition to force the macro to behave in a certain way.

Example

```
OnError (Error@)
     ...
     <your code goes here>
AssertError
     ...
     <perhaps more code goes here>
Label (Error@)
Type ("Hey, bub!  An error occurred!")
```

You can also use the Assert commands to create user-defined conditions, regardless of what WPWin might generate itself. For example, suppose you want to test if a value entered at a **GetNumber** prompt is over-range. Here's one way:

Example

```
OnError (OverRange@)
GetNumber(NumVal;"Enter number from 1 to 10";"Title")
If (NumVal > 10)
     AssertError
EndIf
     ...
     <rest of macro>

Label (OverRange@)
Prompt ("Title";"Value is overrange!";;;)
```

You can use the **AssertCancel** and **AssertNotFound** commands in the same way.

If you have previously used a **CancelOff** or **ErrorOff** command, WPWin will ignore any **AssertCancel** or **AssertError** commands encountered in the macro (there is no "NotFoundOff" command). The default setting is to allow cancel and error conditions; however, you can disallow errors and cancels with **ErrorOff** and **CancelOff**, respectively (this also prevents trapping errors and cancels that WPWin itself generates). You can subsequently allow errors and cancels with **ErrorOn** and **CancelOn**.

Assign

The **Assign** command creates a variable and "assigns" a value to that variable. The syntax for the Assign command is:

Assign (VarName;Expression)

Example

```
Assign (Example; 1000)
     //   Assigns the number 1000 to the variable Example

Assign (MyString;"This is a string")
     //   Assigns "This is a string" to MyString

Assign (Control;TRUE)
     //   Assigns the Boolean value of TRUE to Control

Assign (EmptyString;"")
     //   Creates a string variable EmptyString, but doesn't
          assign anything to it

Assign (Var2Var;Example)
     //   Assigns the contents of Example to the new variable
          Var2Var

Assigns (Example;Example + 1)
     //   Adds 1 to the value already in Example, and assigns the
          new value back to Example
```

Assigning Variables Implicitly

The **Assign** command is the explicit way to assign variables in WPWin macros. An alternative form is the implicit variable assignment, using the following syntax:

Varname:=Expression

Expression can be a valid numerical value, a string, or another variable. Note that there must be both a colon and equals sign between the name of the variable, and the variable expression.

Example

```
Example:=1000
     //   Assigns the number 1000 to the variable Example

MyString:="This is a string"
     //   Assigns "This is a string" to MyString

Control:=TRUE
     //   Assigns the boolean value of TRUE to Control

EmptyString"=""
     //   Creates a string variable EmptyString, but doesn't
          assign anything to it

Var2Var:=Example
     //   Assigns the contents of Example to the new variable
```

```
          Var2Var

Example:=Example + 1
     //   Adds 1 to the value already in Example, and assigns the
          new value back into Example
```

WPWin imposes some severe restrictions on the selection and use of variable names:

- Variable names can contain letters and numbers only. You cannot include spaces, underscores, or symbols.
- The variable name must begin with an alphabetic character -- A through Z. It cannot begin with a number.
- Variable names can be any length, but WPWin uses only the first 50 characters of the name to test for uniqueness. Under normal circumstances you should never have a reason to approach this limit.
- While WPWin macro variables can be up to 50 characters long, WPWin considers only the first seven characters of a merge variable name. Remember this if you have to mix and match macro and merge variables.

Beep

The **Beep** command produces a short tone through the computer's speaker. The command is used without an argument.

Example

```
Prompt ("Title";"You've done it all wrong!";;;)
Beep
Wait (20)
EndPrompt
```

ByteLen

ByteLen counts the number of bytes in a string. If you just want to find out how many individual characters are in a string, use the StrLen command instead. The syntax for **ByteLen** is:

ByteLen (ReturnVal;TestString)

ReturnVal contains the length, in bytes, of TestString.

Example

```
ByteLen (NumBytes;"This is a test")
     //   Returns 14

Assign (MyString;"This is a test")
ByteLen (NumBytes;MyString)
     //   Also returns 14, counting number of bytes in the
          MyString variable
```

In both examples, the NumBytes variable contains the number of bytes in the specified string. Remember that the return value is a number, so if you want to **Type** the number into the document, you have to convert the number to a string first.

Example

```
Assign (MyString;"This is a test")
ByteLen (NumBytes;MyString)
NumStr (sNumBytes;0;NumBytes)
Type (sNumBytes)
     //   Macro Types "14"
```

Counting Extended Bytes

All of the characters in the standard IBM character set (letters, numbers, symbols, Extended IBM Characters) use one byte each. So, a string such as "This is a test" returns a value of 14 when used with **ByteLen**, as "This is a test" contains 14 one-byte characters.

The remaining 1,200+ of WPWin's special extended characters are composed of four bytes each, in the following format:

Byte 1 -- Start extended character code
Byte 2 -- Character set value
Byte 3 -- Character value
Byte 4 -- End extended character code

BytePos

BytePos checks the referenced string for a particular character or group of characters (which is called a substring).

- If the substring is found in the referenced string: The return value is the starting position, in bytes, of the substring.
- If the substring is not found in the referenced string: The return value is zero (0).

Following is the syntax for the **BytePos** command:

BytePos (ReturnVal;SubString;RefString)

RefString is the string to check. SubString is the string to find in RefString. ReturnVal is the starting position, in bytes, of SubString (or 0, if not found).

Example

```
RefString:="This is a test"
SubString:="is"
BytePos (WhereIsIt;SubString;RefString)
     //   WhereIsIt returns 6, as "is" starts at the sixth byte
          in the RefString variable
```

The **BytePos** command counts bytes, not characters. Extended WordPerfect characters are each four bytes (instead of one byte each, as with regular characters). See ByteLen, above, for more information on how WPWin handles extended characters when counting by bytes.

Call

The **Call** command branches a labeled subroutine inside the current macro. After the subroutine is finished, the macro returns to the point immediately after the **Call** command, and continues on. Each **Call**ed subroutine must end with a Return statement, or the macro will end without ever returning to the **Call** command.

The syntax for the Call statement is:

Call (Label@)

Example

```
Call (Loop@)
     ...
     <Rest of macro>
Quit   // The Quit keeps the macro from executing the Loop@ routine a
second time
Label (Loop@)
Type ("This is the subroutine")
Return
```

The examples **Call**s the subroutine that begins with the label Loop@.

The label name used with **Call** should be 15 bytes or less (WPWin uses only the first 15 bytes of a label name to test for uniqueness), and that the label must end with a @ symbol.

CancelOff

CancelOn

The **CancelOff** and **CancelOn** commands are used to disable and enable, respectively, cancel conditions from affecting macro execution. There are two types of cancel conditions: those invoked by the user (press the Cancel key or choose the Cancel button), and those generated internally within a macro using the AssertCancel and ReturnCancel commands. Both **CancelOff** and **CancelOn** are used without an argument.

- With **CancelOff**, the macro will ignore cancel requests by the user, as well as internally generated cancels using the **AssertCancel** and **ReturnCancel** commands.
- With **CancelOn**, the macro accepts cancel requests both by the user, and by those generated internally in the macro.

WPWin automatically assumes cancel trapping is on (equivalent to **CancelOn**) unless you explicitly enter a **CancelOff** command. You can turn cancel on an off as many times within a macro as you wish.

You can specify what should happen if cancel is invoked (assuming the cancel trap is on with **CancelOn**) using the **OnCancel** exception handler command. See OnCancel for more details. If no **OnCancel** exception handler is included in your macro, WPWin will terminate macro macro when a cancel condition occurs.

Case

The **Case** command branches the macro to one of several labeled routines, depending on the match between a test expression, and series of cases. The syntax for **Case** is:

Case (Test;{
    Case1;Label1@;
    Case2;Label2@;
    ...
    Casen;Labeln@};
    DefaultLabel@
)

Notice the syntax: be sure to use all the semi-colons, parentheses, and braces ({ and } characters) that you see, or WPWin will respond with ansyntax error message. The line formatting shown here is optional; you can combine all the arguments on one line if you wish, or use any other formatting style you prefer.

Test is the text expression, and is usually a variable. Test is evaluated against the Case arguments. If there's a match, the macro jumps to the label specified by its Case expression. For example, if Test and Case2 match, the macro jumps to Label2@. You can use as many Cases and Labels as you wish in the Case structure, but remember that every Case must have a matching label.

Example

```
Application (wp;wpwp;default)
GetNumber (NumVar;"Enter 1 or 2";"Title")
Case (NumVar;{
     1;Label1@;
     2;Label2@};
     DefaultLabel@
)
Label (Label1@)
     Type ("You typed 1")
     Quit
Label (Label2@)
     Type ("You typed 2")
     Quit
Label (DefaultLabel@)
     Type ("You didn't type 1 or 2!")
     Quit
```

The **Case** command compares the value in NumVar with the case expressions, in this case 1 and 2.

- If the value in NumVar is 1, the macro branches to the Label1@ label.
- If the value in NumVar is 2, the macro branches to the Label2@ label.
- If the value in NumVar is something other than 1 or 2, the macro branches to the DefaultLabel@ label.

The DefaultLabel@ specifies where the macro should go if there are no matches. You need to include this label even if you want the macro to "fall through" to the next command following the Case structure.

Comparing Against Strings

Quotes are needed when using strings as the case expression.

Example

```
Case (StrVar;{
      "y";Yes@;
      "n";No@};
      Default@
)
```

Matches are case sensitive. If StrVar contains Y, the macro will not find a match even though you have specified a match for y. To accommodate both upper and lower case, enter both variations in the **Case** structure.

Example

```
Case (StrVar;{
      "Y";Yes@;      "y";Yes@;
      "N";No@;
      "n";No@};
      Default@
)
```

Case Call

The **Case Call** command is the same as Case but calls the subroutine identified with the corresponding label. After the subroutine is finished, the macro returns to the point right after the **Case Call** structure and completes the rest of the macro. Each **Case Call**ed subroutine must end with a Return statement.

Note that the **Case Call** command is two separate words. WPWin doesn't recognize the command "CaseCall."

Chain

Chain executes the named macro after the current macro is completed. The **Chain** command uses one argument: the name (and optionally a path) of the macro to execute.

Chain ("macroname")

Note that the name of the macro is enclosed in quotes.

Example

```
Chain ("DOLIST.WCM")
     //   Executes the macro DOLIST.WCM

Assign (PlayMacro;"c:\wpwin\macros\dolist.wcm")
Chain (PlayMacro)
     //   Executes the macro in the PlayMacro variable (note path
          provided with macro name)
```

WPWin insists that before it will **Chain** to a macro, that the macro be compiled first. You need to play the macro at least one if it is not yet compiled (you have never used it before, for example). Or, you can use the Macro Facility program to compile the macro without actually playing it.

Finding Macros

To ensure that your macros are always found, include an explicit directory path so WPWin is sure to locate them. This works fine for macros that are used on a specific system, where you know the directory structure, but it isn't an acceptable approach if the macro may be used on other computers, which may have different directory paths.

In these instances, you can use the GetWPData product function command to return the directory used for storing macro files (as specified in the Location of Files dialog box).

Example

```
GetWPData (MacroPath;MacroPath!)
Chain (MacroPath + "mymacro.wcm")
```

GetWPData returns the full path (with disk drive) of the macro directory. The MacroPath variable is a string.

Chaining in the Middle of a Macro

The **Chain** command is designed so that it executes the named macro when the current macro ends. Therefore, if you include the Chain command somewhere in the middle of the first macro, it won't take effect until the macro ends. This is by design. (If you want the named macro to execute immediately, use the Run command instead.)

In addition, WPWin will only execute the macro specified in the last **Chain** command it encountered. If you have more than one **Chain** command in a macro, only the last one will have any effect.

While WPWin is supposed to behave in these ways, problems can occur if you want to write a macro that **Chain**s to any number of macros, depending on the outcome of some test. Without some careful planning, WPWin either never execute the **Chain**ed macro, or it will **Chain** the wrong one (always the last one it encountered). Fortunately, there's a way out of this mess. Here's an example of how to overcome this limitation:

Example

```
GetString (ChainMac;"1 chains MYMAC; 2 chains YOURMAC";"Title")
If (ChainMac = "1")
     Chain ("MYMAC.WCM")
     Return
EndIf
If (ChainMac = "2")
     Chain ("YOURMAC.WCM")
     Return
EndIf
```

With the Return command added in, WPWin will end the current macro, and execute the **Chain**ed macro immediately. This works because when used outside of a macro nest (using **Run**) or a subroutine, the **Return** command terminates macro execution. Instead of terminating all macro execution, however, WPWin only terminates the current macro, allowing the **Chain**ed macro torun.

DefaultUnits

DefaultUnits specifies a default units of measure to use during macro execution. You can specify a measurement value, without also indicating the unit of measure. The **DefaultUnits** command uses one argument, and only one of the following values (note: these values are not considered strings).

DefaultUnits (UnitOfMeasure)

- Inches
- Centimeters
- Points
- WPUnits

WPWin doesn't not allow you to use a variable in place of the UnitOfMeasure argument. It must be one of the literal values listed above.

Example

```
DefaultUnits (Inches)
     //   Specifies Inches as the default unit of measure

DefaultUnits (Points)      //   Specifies Points as the default unit of
measure
```

If you dont specify a **DefaultUnits**, you will need to append a unit of measure suffix to measurement values applied in your macro (you should not assume a particular unit of measure, or your macro may not work properly). For instance, the **PageMargins** product function command requires that you provide value for the margins you want to set. If you don't set **DefaultUnits**, you need to indicate the unit of measure you are using (" for inches, p for points, c for centimeters, w for WPunits).

DialogAddCheckBox

Use the **DialogAddCheckBox** command to add a check box control to a custom dialog box. The checkbox consists of two portions: the box itself and accompanying text. Click in the box and a checkmark appears. Click again and the checkmark disappears. The checkbox control is used to indicate a yes/no or on/off option.

The **DialogAddCheckBox** command is used with DialogDefine.

One of the parameters of the **DialogAddCheckBox** command is a variable, used to hold the value of the checkbox. The value is 0 (zero) if the box is unchecked, or 1 if the box is checked. You can assign a value to this variable before creating the dialog box that contains the **DialogAddCheckBox** command. Assign a value of 1 to place a check in the box when the dialog appears.

```
DialogAddCheckBox (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Text;
        Variable
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the check box control with a specific dialog box.

ControlID: ID number or name of the check box control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the check box control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the check box control.

Width: The width of the control, in dialog units. The width includes the checkbox itself, and the accompanying text.

Height: The height of the control, in dialog units.

Text: The text that accompanies the check box. The text appears to the right of the check box.

Variable: Return variable that contains the value of the check box (can also be used to "pre-load" a value before the dialog box is created). Values are: 0 for no check; 1 for check.

Ideal Sizes:

Specifications	Width	Height
Box w/ 10 characters	50	8
Box w/ 15 characters	75	8
Box w/ 20 characters	100	8

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="CheckBox test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddCheckBox (
    Box;
    100;
    5;
    5;
    50;
    8;
    "Check 1";
    Check1)

DialogDisplay (Box;1)
NumStr (Check1;;Check1)
Type (Check1)
DialogDestroy (Box)
```

DialogAddColorWheel

Use the **DialogAddColorWheel** command to add a color wheel to a custom dialog box. The wheel consists of two parts: a circle for defining the hue, and a slide control for defining the saturation (intensity).

The **DialogAddColorWheel** command is used with DialogDefine.

One of the parameters of the **DialogAddColorWheel** command is a variable, used to hold the value of the color wheel. The value is a numbner from 0 (black) to 16777215 (white). You can assign a value to this variable before creating the color wheel to display a particular color. The **DialogAddColorWheel** command is most often used to determine the color of fonts.

```
DialogAddColorWheel (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Variable
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the color wheel control with a specific dialog box.

ControlID: ID number or name of the color wheel control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the color wheel control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the color wheel control.

Width: The width of the control, in dialog units. The width includes the entire color wheel control, including color choice wheel and slider bar.

Height: The height of the control, in dialog units.

Variable: Return variable that contains the value of the color wheel (can also be used to "pre-load" a value before the dialog box is created). The value contains the hue and saturation information for the three primary colors, red, blue, and green. To be useful this number must be broken into its constituent colors.

Ideal Size:

Specifications	Width	Height
Standard color wheel	75	75

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Color Wheel test"
ColorWheel:=(128 * 1) + (128 * 256) + (128 * 65536)
```

```
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 2 + 16;
    Title
)

DialogAddColorWheel(
    Box;
    100;
    5;
    5;
    75;
    75;
    ColorWheel
)

DialogDisplay (Box;1)
NumStr (Colorwheel;;Colorwheel)
Type (Colorwheel)
DialogDestroy (Box)
```

DialogAddComboBox

Use the **DialogAddComboBox** command to add a combo list control to a custom dialog box. The combo box is available in three styles, but all three styles do basically the same thing: the box displays one or more lines of text. Each line represents a selectable entry. Click on an entry to select it.

The **DialogAddComboBox** command is used with DialogDefine.

One of the parameters of the **DialogAddComboBox** command is a variable, used to hold the item selected in the combo box. The variable is empty if no item was selected; otherwise the text of the item is returned. You can assign a string value to this variable before creating the dialog box that contains the **DialogAddComboBox** command. This displays an item in the entry field of the combo box when the dialog is first shown.

Items are added to the combo box using the DialogAddListItem command.

```
DialogAddComboBox (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Variable;
        DataLimit
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the combo box control with a specific dialog box.

ControlID: ID number or name of the combo box control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the combo box control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the combo box control.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units. The height includes the text entry field plus the drop down list (the drop down list is always displayed when using the Standard style).

Style: Specifies the style of the box: Standard, Dropdown, or Droplist, as defined in the table below.

Variable: Return variable that contains the text of the selected item.

DataLimit: Specifies the maximum number of characters allows in the edit field box.

Style Definitions:

Style	Number	Description
Standard	0	Appears as a dropdown list already opened. You can enter

		text into the edit box to "speed search" through the available items.
Dropdown	8	Appears as edit box; click on box to drop down list of all items. You can enter text into the edit box to "speed search" through the available items.
DropList	16	Same as above, but the control has a 3-D look, and you cannot enter text into the edit box.

Ideal Sizes:

Specifications	Width	Height
20 chars max	75	32
35 chars max	125	32

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

BoxVar:="This is also a test"
Title:="Combo Box test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddComboBox(
    Box;
    100;
    5;
    5;
    100;
    32;
    16;
    BoxVar;
    35
)

DialogAddListItem(
    Box;
    100;
    "This is a test"
)

DialogAddListItem(
    Box;
    100;
    "This is also a test"
)

DialogAddListItem(
    Box;
```

```
    100;
    "This is the third test"
)

DialogDisplay (Box;1)
Type (BoxVar)
DialogDestroy (Box)
```

DialogAddCounter 

Use the **DialogAddCounter** command to add a counter control to a custom dialog box. The counter control consists of two parts: a text entry box and up/down buttons ("spin buttons"). A number can be typed directly into the edit box, ot the up/down buttons can be pressed with the mouse to change the value in the edit box.

The **DialogAddCounter** command is used with DialogDefine.

One of the parameters of the **DialogAddCounter** command is a variable, used to hold the number contained in the edit box. You can assign a numeric value to this variable before creating the dialog box that contains the **DialogAddCounter** command. This displays the value in the counter control when when the dialog is first shown.

The **DialogAddCounter** command lets you define the minimum and maximum values that can be entered into the edit box, as well as a "step" value. The step value determines how the number in the edit box is incrememented each time you press the up/down buttons on the control. In addition, you can specify the type of value used with the counter control, such as numbers, centimeters, inches, percent, or points.

```
DialogAddCounter (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Format;
        Variable;
        MinValue;
        MaxValue;
        StepValue
)
```

Parameters:

DialogID: ID number or name of the parent dialog box. This references the counter control with a specific dialog box.

ControlID: ID number or name of the counter control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the counter control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the counter control.

Width: The width of the control, in dialog units. The width includes the up/down buttons

Height: The height of the control, in dialog units. The height includes the up/down buttons.

Format: Specifies the type of numeric value used with the control, as defined in the table below.

Variable: Return variable that contains the value of the counter.

MinValue: Specifies the minimum value that can be represented in the counter.

MaxValue: Specifies the maximum value that can be represented in the counter.

StepValue: Specifies the increment value when the up/down buttons of the counter control are pressed.

Format Definitions:

Format	Number
Normal	0
WPUnits "w"	1
Points	2
Centimeters	3
Inches ".0"	4
Percent "%"	5
Inches	6
Inches "i"	7

Ideal Sizes:

Specifications	Width	Height
Box for 1 digit/chars	25	15
Box for 2 digits/chars	30	15
Box for 4 digits/chars	35	15
Box for 6 digits/chars	45	15
Box for 8 digits/chars	55	15

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Counter test"
Box:="MyBox"
Counter1:=1
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddCounter (
    Box;
    100;
    5;
    5;
    30;
    15;
    0;
    Counter1;
    1;
    10;
    1
)

DialogDisplay (Box;1)
NumStr (Counter1;;Counter1)
```

```
Type (Counter1)
DialogDestroy (Box)
```

DialogAddEditBox

Use the **DialogAddEditBox** command to add an edit box control to a custom dialog box. The edit box is available in numerous styles that define the look and features of the control. The edit box can be designed to hold one line of text, or many

The **DialogAddEditBox** command is used with DialogDefine.

One of the parameters of the **DialogAddEditBox** command is a variable, used to hold the text entered into the box. The variable is empty if the box is empty. You can assign a string value to this variable before creating the dialog box that contains the **DialogAddEditBox** command. This displays text in the edit box when the dialog is first shown.

```
DialogAddEditBox (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Variable;
        DataLimit
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the edit box control with a specific dialog box.

ControlID: ID number or name of the edit box control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the edit box control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the edit box control.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

Style: Specifies the style of the box, as defined in the table below.

Variable: Return variable that contains the text entered into the edit box, if any.

DataLimit: Specifies the maximum number of characters allowed in the edit box.

Style Definitions:

Style	Number	Description
Left	1	Left justified text
Right	2	Right justified text when used with WPChars
Center	4	Centered text when used with WPChars
VScroll	8	Adds a vertical scroll bar to the text box (used when the box displays multiple lines)

HScroll	16	Adds a horizontal scroll bar to the text box (used when the box displays more characters than will fit in the edit field)
WPChars	32	Specifies WP characters allowed
Multiline	64	Specifies box allows multiple lines
Uppercase	128	Forces all entry to upper case
Lowercase	256	Forces all entry to lower case
Password	512	Types "*" for each character entered
WordWrap	1024	Wraps line if string exceeds the width of the edit box
AllSoftRet	2048	When WordWrap is enabled, soft returns at line breaks are formatted into the output
Attributes	4096	Allows character attributes (bold, underline, etc.)
NoTabs	8192	Removes tabs from the output
NoWPChar	16384	Disallows display of WP Characters dialog box

Ideal Sizes:

Specifications	Width	Height
Single line; 10 chars	40	12
Single line; 15 chars	55	12
Single line; 20 chars	85	12
Single line; 40 chars	175	12
Two lines	x	24
Three lines	x	36
Four lines	x	48

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

EditVar:="This is a test"
Title:="Edit Box test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddEditBox(
    Box;
    100;
    5;
    5;
    125;
    30;
    0;
    EditVar;
    70
)

DialogAddText(
    Box;
    "Text1";
    15;
```

```
    45;
    100;
    30;
    0;
    "This is a test"
)

DialogDisplay (Box;1)
Type (EditVar)
DialogDestroy (Box)
```

DialogAddFilenameBox

Use the **DialogAddFilenameBox** command to add a filename box control to a custom dialog box. The filename box is composed of two parts: a text entry field for manually entering the filename, plus a directory (or "browse") button. Press the directory button with the mouse and an Open File dialog box appears.

The **DialogAddFilenameBox** command is used with DialogDefine.

One of the parameters of the **DialogAddFilenameBox** command is a variable, used to hold the selected filename, if any. The variable is empty if no filename was selected; otherwise the variable contains the name of the file, with full path.. You can assign a string value to this variable before creating the dialog box that contains the **DialogAddFilenameBox** command. This displays an item in the entry field of the filename box when the dialog is first shown.

```
DialogAddFilenameBox (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Variable;
        DefDir;
        Template
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the filename box control with a specific dialog box.

ControlID: ID number or name of the filename box control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the filename box control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the filename box control.

Width: The width of the control, in dialog units. The width includes the "browse" button

Height: The height of the control, in dialog units. The height includes the "browse" button

Style: Specifies options, as defined in the table below.

Variable: Return variable that contains the name and path of the selected file.

DefDir: Specifies the default directory.

Template: Specifies the initial filename pattern, such as *.* (for all files), *.WCM (for WPWin macros), etc.

Style Definitions:

Style	Number	Description

DirPath	1	Only directories are selected in the Select Directory dialog box that appears when the "browse" button is pushed. When the DirPath style is not used, an Open File dialog box appears instead.
NoValidate	2	Turns off error checking for non-existent directories or files.

Ideal Sizes:

Specifications	Width	Height
Box for 12 chars	60	15
Box for 25 chars	100	15
Box for 40 chars	175	15

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Var:="chime.wav"
Title:="Filename Box test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 2 + 16;
    Title
)

DialogAddFilenameBox(
    Box;
    100;
    5;
    5;
    125;
    15;
    0;
    Var;
    "C:\WINDOWS\";
    "*.WAV"
)

DialogDisplay (Box;1)

If (MacroDialogResult = 1)
    DLLLoad (MMSound;"MMSYSTEM.DLL")
    DLLCall (MMSound; "sndPlaySound"; sndSound:BOOL;{
        Var;
    LOWORD (1)})
    DLLFree (MMSound)
EndIf

DialogDestroy (Box)
```

DialogAddFrame

Use the **DialogAddFrame** command to add a frame border to a custom dialog box. The frame is available in a number or colors, with and without a filling shade. The frame is a "static" control and is not active when the dialog box is displayed. It is mostly used to group other controls to show a common relationship.

The **DialogAddComboBox** command is used with DialogDefine.

```
DialogAddFrame (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the frame with a specific dialog box.

ControlID: ID number or name of the frame. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the frame.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the frame.

Width: The width of the frame, in dialog units.

Height: The height of the frame, in dialog units.

Style: Specifies the style of the frame, as defined in the table below.

Style Definitions:

Style	Number	Description
Frame border	0	Colors frame border
Frame interior	4	Colors inside of frame
Black color	0	Black border or interior
Gray color	1	Gray border or interior
White color	2	White border or interior

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Frame test"
Box:="MyBox"

DialogDefine (
```

```
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddFrame (
    Box;
    100;
    5;
    5;
    100;
    100;
    0
)

DialogDisplay (Box;1)
DialogDestroy (Box)
```

DialogAddGroupBox

Use the **DialogAddGroupBox** command to add a group box to a custom dialog box. The group box is a "static" control and and is not active when the dialog box is displayed. It is mostly used to group other controls to show a common relationship. In addition, when used with radio button controls, the group restricts the choice to only one of the buttons in the box. Placed outside a group box, a radio button can be turned on or off irrespective of other radio buttons in the dialog box,

The **DialogAddGroupBox** command is used with DialogDefine.

```
DialogAddGroupBox (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Title)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the group box control with a specific dialog box.

ControlID: ID number or name of the group box control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the group box control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the group box control.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units

Title: Specifies the title of the group box. The title appears centered in the top border of the box.

Ideal Sizes:

Specifications	Width	Height
Box for 2 radio buttons	x	40
Box for 3 radio buttons	x	55
Box for 4 radio buttons	x	65
Box for 6 radio buttons	x	90
Box for 8 radio buttons	x	115
Box for 10 radio buttons	x	145

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")
Title:="Pick a Button"
Box:="MyBox"
LeftEdge:=15
RBWidth:=40
RBHeight:=8
```

```
VPos:=20
Spacing:=5
Tab:=20

DialogDefine (
    Box;
    50;
    50;
    115;
    180;
    1 + 16;
    Title
)

DialogAddGroupBox (
    Box;
    "Group";
    LeftEdge - 10;
    VPos - 12;
    60;
    55;
    "Bullet"
)

DialogAddRadioButton (
    Box;
    "Radio1";
    LeftEdge;
    VPos;
    RBWidth;
    RBHeight;
    "Radio1";
    Radio1
)
VPos:= VPos + RBHeight + Spacing
DialogAddRadioButton (
    Box;
    "Radio2";
    LeftEdge;
    VPos;
    RBWidth;
    RBHeight;
    "Radio2";
    Radio2
)
VPos:= VPos + RBHeight + Spacing
DialogAddRadioButton (
    Box;
    "Radio3";
    LeftEdge;
    VPos;
    RBWidth;
    RBHeight;
    "Radio3";
    Radio3
)
DialogDisplay (Box;1)
```

DialogDestroy (Box)

DialogAddHLine

Use the **DialogAddHLine** command to add a horizontal line to a custom dialog box. The line is a "static" control and and is not active when the dialog box is displayed. It is mostly used to separate other controls and for aesthetic effect.

The **DialogAddHLine** command is used with DialogDefine.

```
DialogAddHLine (
        DialogID;
        ControlID;
        HPos;
        VPos;
        LineLength
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the line with a specific dialog box.

ControlID: ID number or name of the line. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the line, in dialog units. The distance is from the left edge of the dialog box to the left edge of the line.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the line.

LineLength: The length of the line, in dialog units.

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="H Line Example"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddVLine(
    Box;
    101;
    5;
    5;
    175
)
```

```
DialogAddVLine(
    Box;
    102;
    175;
    5;
    175
)

DialogAddHLine(
    Box;
    103;
    10;
    5;
    160
)

DialogAddHLine(
    Box;
    104;
    10;
    180;
    160
)

DialogDisplay (Box;1)
DialogDestroy (Box)
```

DialogAddHotSpot

Use the **DialogAddHotSpot** command to add a "hot spot" control to a custom dialog box. The hot spot serves as an "invisible" button that can be be selected just like a push button. Most often, the hot spot is placed over an icon (using the DialogAddIcon command), or static text (using the DialogAddText command).

The **DialogAddHotSpot** command is used with DialogDefine.

```
DialogAddHotSpot (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
)
```

Parameters:

DialogID: ID number or name of the parent dialog box. This references the hot spot control with a specific dialog box.

ControlID: ID number or name of the hot spot control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the hot spot control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the hot spot control.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

Style: Specifies the style of the hot spot, as defined in the table below.

Style Definitions:

Style	Number	Description
Click	1	Click once on hot spot to select it
DoubleClick	2	Double click on hot spot to select it

Ideal Sizes:

Specifications	Width	Height
Hot spot for small icon (16x16)	10	10
Hot spot for large icon (32x32)	20	20

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Hot Spot test"
Box:="MyBox"
```

```
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 2 + 16;
    Title
)

DialogAddHotSpot(
    Box;
    101;
    5;
    5;
    22;
    23;
    1
)


DLLLoad (MCR; "WPCMCR.DLL")
DialogAddIcon (
    Box;
    102;
    7;
    7;
    20;
    20;
    "wpstamp";
    MCR
)
DialogDisplay (Box;1)
DLLFree (MCR)
NumStr (Result;;MacroDialogResult)
Type (Result)
DialogDestroy (Box)
```

DialogAddIcon

Use the **DialogAddIcon** command to add an icon graphic to a custom dialog box. The icon is a "static" control and and is not active when the dialog box is displayed. It is mostly used for aesthetic effect or to add graphics to the dialog box. However, the icon can be used with the DialogAddHotSpot command to make the image function like a graphic push button.

The **DialogAddIcon** command is used with DialogDefine.

Icons for the **DialogAddIcon** command must be contained in a dynamic link library (DLL) or Windows EXE file. The WPCMCR.DLL file, included with WPWin, contains several icons you can use. The WPWPICON.DLL file, provided on the Applications Disks accompanying WordPerfect for Windows Power Tools, provides about 75 general and special purpose icons that you can use with the DialogAddIcon command.

```
DialogAddIcon (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height
        IconName;
        ModuleLink
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the icon control with a specific dialog box.

ControlID: ID number or name of the icon. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the icon.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the icon.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

IconName: The name of the icon resource contained in the DLL or EXE file. The icon must be named; an icon ID number cannot be used

ModuleLink: The module or library handle of the DLL or EXE file that contains the icon. When using DLL files you can use the WPWin DLLLoad command to obtain the module link handle.

Ideal Sizes:

Specifications	Width	Height
Small icon (16 by 16 pixels)	10	10
Large icon (32 by 32 pixels)	20	20

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Title"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 2 + 16;
    Title
)

DialogAddHotSpot(
    Box;
    101;
    5;
    5;
    22;
    23;
    1
)


DLLLoad (MCR; "WPCMCR.DLL")
DialogAddIcon (
    Box;
    102;
    7;
    7;
    20;
    20;
    "wpstamp";
    MCR
)
DialogDisplay (Box;1)
DLLFree (MCR)
NumStr (Result;;MacroDialogResult)
Type (Result)
DialogDestroy (Box)
```

DialogAddListBox

Use the **DialogAddListBox** command to add a list box control to a custom dialog box. The list box is available in two styles, but both are used to display one or more lines of text. Each line represents a selectable entry. Click on an entry to select it.

The **DialogAddListBox** command is used with DialogDefine.

One of the parameters of the **DialogAddListBox** command is a variable, used to hold the item selected in the list box. The variable is empty if no item was selected; otherwise the text of the item is returned. You can assign a string value to this variable before creating the dialog box that contains the **DialogAddListBox** command. This displays an item in the entry field of the list box when the dialog is first shown.

Items are added to the list box using the DialogAddListItem command.

```
DialogAddComboBox (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Variable
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the list box control with a specific dialog box.

ControlID: ID number or name of the list box control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the list box control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the list box control.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

Style: Specifies the style of the box, either Sort or NameSeach, as defined in the table below.

Variable: Return variable that contains the value of the list box (can also be used to "pre-load" a value before the dialog box is created).

Style Definitions:

Style	Number	Description
Sort	1	Sorts items before displaying them
NameSearch	2	Allows name search lookup of text entries

Ideal Sizes:

Specifications	Width	Height
20 chars max	75	*
35 chars max	125	*

* Allow 10 dialog units per text line

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="ListBox test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddListBox(
    Box;
    100;
    5;
    5;
    100;
    32;
    0;
    ListBoxVar
)

DialogAddListItem(
    Box;
    100;
    "This is a test"
)

DialogAddListItem(
    Box;
    100;
    "This is also a test"
)

DialogAddListItem(
    Box;
    100;
    "This is the third test"
)

DialogDisplay (Box;1)
Type (ListBoxVar)
DialogDestroy (Box)
```

DialogAddListItem

Use the **DialogAddListItem** command to add text entires to combo boxes, list boxes, and popup push buttons. Each item in a list represents a selectable entry. Click on an entry to select it.

The **DialogAddListItem** command is used with DialogDefine.

```
DialogAddListItem (
        DialogID;
        ControlID;
        Text
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the list items with a specific dialog box.

ControlID: ID number or name of the parent list control (combo, list, or popup button).

Text: Text for the list entry

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Add List Item test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddPopUpButton(
    Box;
    100;
    5;
    5;
    100;
    32;
    PopUpButtonVar
)

DialogAddListItem(
    Box;
    100;
    "This is a test"
)

DialogAddListItem(
```

```
    Box;
    100;
    "This is also a test"
)

DialogAddListItem(
    Box;
    100;
    "This is the third test"
)

DialogDisplay (Box;1)
Type (PopUpButtonVar)
DialogDestroy (Box)
```

DialogAddPopUpButton

Use the **DialogAddPopupButton** command to add a popup button control to a custom dialog box. The popup button normally appears as a push button. Select the button and a popup list appears. Each line in the list represents a selectable entry. Click on an entry to select it.

The **DialogAddPopUpButton** command is used with DialogDefine.

One of the parameters of the **DialogAddPopUpButton** command is a variable, used to hold the item selected in the button. The variable is empty if no item was selected; otherwise the text of the item is returned. You can assign a string value to this variable before creating the dialog box that contains the **DialogAddPopUpButton** command. This displays an item in the entry field of the popup button when the dialog is first shown.

Items are added to the popup button using the DialogAddListItem command.

```
DialogAddPopUpButton (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Variable
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the popup button control with a specific dialog box.

ControlID: ID number or name of the popup button control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the popup button control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the popup button control.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

Variable: Return variable that contains the value of the popup button (can also be used to "pre-load" a value before the dialog box is created).

Ideal Sizes:

Specifications	Width	Height
20 chars	75	15
35 chars	125	15

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")
```

```
Title:="Popup Button test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddPopUpButton(
    Box;
    100;
    5;
    5;
    100;
    32;
    PopUpButtonVar
)

DialogAddListItem(
    Box;
    100;
    "This is a test"
)

DialogAddListItem(
    Box;
    100;
    "This is also a test"
)

DialogAddListItem(
    Box;
    100;
    "This is the third test"
)

DialogDisplay (Box;1)
Type (PopUpButtonVar)
DialogDestroy (Box)
```

DialogAddPushButton

Use the **DialogAddPushButton** command to add a pushbutton control to a custom dialog box. The button is used to select an option. The size of the button, as well as the text on it, are user-definable.

The **DialogAddPushButton** command is used with DialogDefine.

```
DialogAddPushButton (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Text)
```

Parameters:

DialogID: ID number or name of the parent dialog box. This references the pushbutton control with a specific dialog box.

ControlID: ID number or name of the pushbutton control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the button.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the button.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

Style: Specifies the style of the button, as defined in the table below.

Text: Specifies the text that appears on the face of the button.

Style Definitions:

Style	Number	Description
Default	1	Sets button as the default (only one button in the dialog box should be set as the default)

Ideal Sizes:

Specifications	Width	Height
*Matches defaullt OK/Cancel	32	13
10 chars	50	13
15 chars	70	13
20 chars	90-	13

* This size matches the default OK and Cancel buttons that can be automatically added to the dialog box when defining a style with the DialogDefine command.

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Push Button test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 2 + 16;
    Title
)

DialogAddPushButton (
    Box;
    100;
    80;
    185;
    32;
    13;
    0;
    "Button1"
)

DialogAddPushButton (
    Box;
    101;
    156;
    165;
    32;
    13;
    0;
    "Button2"
)


DialogDisplay (Box;1)
NumStr (Result;;MacroDialogResult)
Type (Result)
DialogDestroy (Box)
```

DialogAddRadioButton

Use the **DialogAddRadioButton** command to add a radio button control to a custom dialog box. Radio buttons are most often used to select one item among several. When the radio button is placed inside a group box (using the DialogAddGroupBox command), only one radio button can be selected at a time.

The **DialogAddRadioButton** command is used with DialogDefine.

One of the parameters of the **DialogAddRadioButton** command is a variable, used to hold the value of the radio button. If the button is not selected, the value is 0; if the button is selected, the value is 1.

```
DialogAddRadioButton (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Text;
        Variable
)
```

Parameters:

DialogID: ID number or name of the parent dialog box. This references the radio button control with a specific dialog box.

ControlID: ID number or name of the radio button control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the radio button control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the radio button control.

Width: The width of the control, in dialog units. The width includes the radio button "bullseye."

Height: The height of the control, in dialog units. The height includes the radio button "bullseye."

Text: The text for the radio button.

Variable: Return variable that contains the value of the radio button (can also be used to "pre-load" a value before the dialog box is created). Values are: 0 for button not active; 1 for active.

Ideal Sizes:

Specifications	Width	Height
10 chars	45	8
15 chars	65	8
20 chars	90	8

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")
Title:="Pick a Button"
Box:="MyBox"
```

```
LeftEdge:=15
RBWidth:=40
RBHeight:=8
VPos:=20
Spacing:=5
Tab:=20

DialogDefine (
    Box;
    50;
    50;
    115;
    180;
    1 + 16;
    Title
)

DialogAddGroupBox (
    Box;
    "Group";
    LeftEdge - 10;
    VPos - 12;
    60;
    55;
    "Bullet"
)

DialogAddRadioButton (
    Box;
    "Radio1";
    LeftEdge;
    VPos;
    RBWidth;
    RBHeight;
    "Radio1";
    Radio1
)

VPos:= VPos + RBHeight + Spacing

DialogAddRadioButton (
    Box;
    "Radio2";
    LeftEdge;
    VPos;
    RBWidth;
    RBHeight;
    "Radio2";
    Radio2
)

VPos:= VPos + RBHeight + Spacing

DialogAddRadioButton (
    Box;
    "Radio3";
    LeftEdge;
```

```
    VPos;
    RBWidth;
    RBHeight;
    "Radio3";
    Radio3
)


DialogDisplay (Box;1)
If (Radio1=1) Type ("Radio1 Selected") EndIf
If (Radio2=1) Type ("Radio2 Selected") EndIf
If (Radio3=1) Type ("Radio3 Selected") EndIf
DialogDestroy (Box)
```

DialogAddScrollBar

Use the **DialogAddScrollBar** command to add a scroll bar control to a custom dialog box. The scroll bar can be either vertical or horizontal.

The **DialogAddScrollBar** command is used with DialogDefine.

One of the parameters of the **DialogAddScrollBar** command is a variable, used to hold the numeric value of the scroll bar.

```
DialogAddScrollBar (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Variable;
        MinValue;
        MaxValue
)
```

Parameters:

DialogID: ID number or name of the parent dialog box. This references the scroll bar control with a specific dialog box.

ControlID: ID number or name of the scroll bar control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the scroll bar control.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the scroll bar control.

Width: The width of the control, in dialog units. The width includes the arrows and thumb box of the scroll bar.

Height: The height of the control, in dialog units. The height includes the arrows and thumb box of the scroll bar.

Style: Specifies the style of the box, as defined in the table below.

Variable: Return variable that contains the value of the scroll bar counter (can also be used to "pre-load" a value before the dialog box is created).

MinValue: The minimum value of the scroll bar counter.

MaxValue: The maximum value of the scroll bar counter.

Style Definitions:

Style	Number	Description
Left/Top	1	Positions thumb box at left (or top) of bar
Right/Bottom	2	Positions thumb box at right (or bottom) or bar

VScroll	8	Specifies vertical scroll bar
HScroll	16	Specifies horizontal scroll bar

Ideal Sizes (given for horizontal bar):

Specifications	Width	Height
Short bar	75	16
Medium bar	100	16
Long bar	150	16

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Scrollbar test"
Box:="MyBox"

DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddscrollBar (
    Box;
    100;
    5;
    5;
    100;
    16;
    1+16;
    ScrollBar;
    0;
    10
)

DialogDisplay (Box;1)
NumStr (ScrollBar;;ScrollBar)
Type (ScrollBar)
DialogDestroy (Box)
```

DialogAddText

Use the **DialogAddText** command to add text to a custom dialog box. The text is "static" and is not active. It is mostly used to add explanatory text to the dialog box.

The **DialogAddText** command is used with DialogDefine.

```
DialogAddText (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Text
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the text with a specific dialog box.

ControlID: ID number or name of the text. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the text.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the text.

Width: The width of the control, in dialog units.

Height: The height of the control, in dialog units.

Style: Specifies the style of the box, as defined in the table below.

Text: Specifies the text.

Style Definitions:

Style	Number	Description
Left	1	Left justified
Right	2	Right justified (when used with WPChars)
Center	4	Center justified (when used with WPChars)
RecessedBox	8	Sets text in a 3-D recessed box
ShadowBox	16	Sets text in a 3-D shadow box
WPChars	32	Enables display of WP characters

Ideal Sizes:

Specifications	Width	Height
10 chars	30	8
15 chars	50	8
20 chars	60	8
30 chars	110	8
40 chars	135	8

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="AddText test"
Box:="MyBox"

DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddtext (
    Box;
    "Text1";
    5;
    5;
    100;
    8;
    1;
    "This is a test"
)
DialogDisplay (Box;1)
DialogDestroy (Box)
```

DialogAddViewer

Use the **DialogAddViewer** command to add a veriwe control to a custom dialog box. The viewer displays a box that contains the text of a specified file. Scroll bars appear on the bottom and right borders of the viewer so that the text can be scrolled within the box.

The **DialogAddViewer** command is used with DialogDefine.

```
DialogAddViewer (
        DialogID;
        ControlID;
        HPos;
        VPos;
        Width;
        Height;
        Filename
)
```

Parameters:

DialogID: ID number or name of the parent dialog box. This references the viewer control with a specific dialog box.

ControlID: ID number or name of the viewer control. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the control, in dialog units. The distance is from the left edge of the dialog box to the left edge of the viewer box.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the viewer box.

Width: The width of the control, in dialog units. The width includes the vertical scroll bar that appears on the right side of the viewer box.

Height: The height of the control, in dialog units. The height includes the horizontal scroll bar that appears on the bottom of the viewer box.

Filename: Specifies the file to view.

Ideal Sizes:

Specifications	Width	Height
Small viewer box; narrow	100	100
Small viewer box; wide	200	100
Large viewer box; narrow	150	200
Large viewer box; wide	200	200

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Viewer test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
```

```
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddViewer(
    Box;
    100;
    5;
    5;
    184;
    175;
    "C:\windows\wpwp.ini"
)

DialogDisplay (Box;1)
DialogDestroy (Box)
```

DialogAddVLine

Use the **DialogAddVLine** command to add a vertical line to a custom dialog box. The line is a "static" control and and is not active when the dialog box is displayed. It is mostly used to separate other controls and for aesthetic effect.

The **DialogAddVLine** command is used with DialogDefine.

```
DialogAddVLine (
        DialogID;
        ControlID;
        HPos;
        VPos;
        LineLength
)
```

Parameters:
DialogID: ID number or name of the parent dialog box. This references the line with a specific dialog box.

ControlID: ID number or name of the line. Be sure to use a different ID for each control in the dialog box.

HPos: Horizontal position of the line, in dialog units. The distance is from the left edge of the dialog box to the left edge of the line.

VPos: Vertical position of the control, in dialog units. The distance is from the top of the dialog box to the top of the line.

LineLength: The length of the line, in dialog units.

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="V Line test"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 16;
    Title
)

DialogAddVLine(
    Box;
    101;
    5;
    5;
    175
)
```

```
DialogAddVLine(
    Box;
    102;
    175;
    5;
    175
)

DialogAddHLine(
    Box;
    103;
    10;
    5;
    160
)

DialogAddHLine(
    Box;
    104;
    10;
    180;
    160
)

DialogDisplay (Box;1)
DialogDestroy (Box)
```

DialogDefine

The **DialogDefine** command creates a dialog box template. Individual controls -- such as text, push buttons, and check boxes --- can then be added to the dialog box. You can define as many dialog boxes as you want per macro, but only one dialog box can be displayed at any one time.

The **DialogDefine** command merely creates the dialog box in memory. To show the box you must use the DialogDisplay command.

When done with the dialog box, use the DialogDestroy command to remove the dialog box from memory.

```
DialogDefine (
        DialogID;
        HPos;
        VPos;
        Width;
        Height;
        Style;
        Caption
)
```

Parameters:
DialogID: Specifies a unique ID number or text for the dialog box. Each dialog box in your macro should use a different ID.

HPos: Horizontal position of the dialog box, in either dialog units or a percentage of the screen. The distance is from the left edge of the screen to the left edge of the dialog box.

VPos: Vertical position of the dialog box, in either dialog units or a percentage of the screen. The distance is from the top of the screen to the top of the dialog box.

Width: The width of the dialog box, in dialog units. The width includes the frame around the box (when the box is displayed with a frame).

Height: The height of the dialog box, in dialog units. The height includes the frame around the box as well as the caption bar (when the box is displayed with a frame and caption).

Style: Specifies the style of the dialog box, as defined in the table below.

Caption: Specifies the title of the dialog box, displayed in the caption bar.

Style Definitions:

Style	Number	Description
DialogUnits	0	Measurement of box depends on the Windows font used for dialog boxes
Modal	0	Dialog box must be dismissed before continuing
NoButton	0	No default buttons displayed
OK	1	OK button displayed
Cancel	2	Cancel button displayed
Modeless	8	Dialog box does not need to be dismissed before doing other tasks in WPWin
Percent	16	Sets measurement as a percentage of the screen
NoFrame	32	Removes the frame around the dialog box
Sizeable	128	Enables dialog box to be resized

NoCaption	256	Removes the caption (title) from the dialog box

Ideal Sizes:

Specifications	Width	Height
Small dialog box	100	140
Medium dialog box	175	200
Large dialog box	300	275

Example:

```
Application (wp;wpwp;default;"wpwpUS.wcd")

Title:="Dialog box only"
Box:="MyBox"
DialogDefine (
    Box;
    50;
    50;
    200;
    220;
    1 + 2 + 16;
    Title
)
DialogDisplay (Box;1)
DialogDestroy (Box)
```

DialogDestroy 

The **DialogDestroy** command removes the dialog box, created with DialogDefine, from memory. Be sure to use **DialogDestory** for every dialog box you define.

```
DialogDestory (
        DialogID
)
```

Parameters:
DialogID: Specifies the ID number or text of the dialog box to remove from memory

DialogDisplay

Use the **DialogDisplay** command to display a dialog box created with DialogDefine. The **DialogDisplay** command should be placed after all the controls for the dialog box have been defined.

```
DialogDisplay (
        DialogID;
        ControlID
)
```

Parameters:
DialogID: Specifies the ID number or text of the dialog box to display.

ControlID: Specifies the ID number or text of the control that is to have the initial "focus" when the dialog box first displays. The focus sets the default selection of the control. To set the OK button as the default (if shown), use a ControlID of 1; to set the Cancel button as the default (if shown), use a ControlID of 2. Or, use the unique ControlID of a custom control you have added to the dialog box.

DLLCall

DLLFree

DLLLoad

The DLL commands are used to access functions contained in dynamic link library (DLL) files. DLL functions greatly enhance the capability of WPWin macros. With care, you can use them to tap into the high-level functions built into Windows itself. You can also use a number of functions built into WordPerfect for Windows.

The DLL commands should be used by advanced WPWin macro writers only, as misuse can cause Unrecoverable Application Errors (Windows 3.0), or General Protection Faults (Windows 3.1). They are are covered in-depth in Chapter 15, "Using DLL Macro Functions" of **WordPerfect for Windows Power Tools**. There, you will learn how to apply the DLL commands and use them with several hundred Windows and WPWin functions, plus more than four dozen functions exclusive with this book.

Else

The **Else** command is used with the **If** and **EndIf** commands. It indicates what should happen should the **If** statement is FALSE.

- Should the **If** statement evaluate to TRUE, the commands between **Else** and **EndIf** are ignored.
- Should the **If** statement evaluate to FALSE, the commands between the If and Else are ignored, and the macro executes those commands between the **Else** and the **EndIf**

See the description of If for an example of how the **Else** command is used.

EndApp

Use **EndApp** to tell WPWin that you're done using the application defined in the Application command. Any subsequent product commands in the macro for that application will not be valid, and could cause a syntax error.

The **EndApp** command does not use an argument.

EndFor

The **EndFor** command marks the end of a **For** or **ForEach** loop. See the For and ForEach entries for a description of how to use **EndFor**. The **EndFor** command is used without an argument.

EndIf

EndIf marks the end of an **If** structure. See the description of If for an example of how the **EndIf** command is used.

EndPrompt

The **EndPrompt** command clears the message box displayed by the **Prompt** command. See the description of Prompt for an example of how to use the **EndPrompt** command. The **EndPrompt** command is used without an argument.

EndWhile

EndWhile end a **While** loop. See the While command for a description of how to use **EndWhile**.
EndWhile is used without an argument.

ErrorOff

ErrorOn

The **ErrorOff** and **ErrorOn** commands are used to disable and enable, respectively, error conditions from affecting macro execution. Error conditions can be caused by a system fault when running the macro, or by the AssertError and ReturnError commands. Both **ErrorOff** and **ErrorOn** are used without an argument.

- With **ErrorOff**, the macro will ignore "non-fatal" error conditions, as well as internally generated errors using the **AssertError** and **ReturnError** commands.
- With **ErrorOn**, the macro accepts error conditions both generated by WPWin, and those generated internally in the macro with **AssertError** and **ReturnError**.

A "non-fatal" error is one that do not require intervention by the user, and does not to cause a major disruption in the operation of WPWin or the macro. Errors in macro programming are considered "fatal," and the **ErrorOn/ErrorOff** commands have no affect with these (programming errors include trying to **Type** a numeric value).

Additionally, serious system errors, such as Unrecoverable Application Errors (UAEs, in Windows 3.0) or General Protection Faults (GPFs, in Windows 3.1), cannot be excluded using **ErrorOn**/**ErrorOff**.

For

The For command repeats a loop a specified number of times. The loop itself normally consists of one or more commands as a repeating routine. This routine is enclosed between the **For** command and a terminating **EndFor** command.

The **For** command uses four arguments: Var, InitVal, Test, and Increment, with the following syntax:

For (Var;InitVal;Test;Increment)

- Var is a unique variable name that contains the "loop counter,". When the **For** loop is first run, Var contains the InitVal number.
- InitVal is the starting value of the **For** loop, and is often 0 or 1. But it can be any number.
- Test is the expression used by the **For** command to control the number of iterations of the loop. As long as the Test expression is TRUE, the **For** loop continues. When the Test expression proves FALSE, the **For** loop ends.
- Increment indicates how you want the **For** loop to count, by 1s, 2s, 5s, 10s, etc.

Here's an example of a **For** loop that counts from 1 to 10, stepping 1 digit at a time. At each iteration the macro inserts another hard return (using the **HardReturn** product function command). When the macro is done there will be 10 hard returns in the current document.

Example

```
For (Count;1;Count<=10;Count+1)
     HardReturn()
EndFor
```

Count is the variable name for the Var argument. The For loop starts out with 1. The Test expression is

Count<=10

which reads "Count is less than or equal to 10." As long as this expression is TRUE, the **For** loop continues. The Increment argument is also an expression, and uses the Count variable to increment the **For** loop by 1 for each iteration.

ForEach

ForEach is a unique looping command. It uses a variable as a temporary placeholder and allows you to assign a series of expressions to that variable with each cycle of the loop. The syntax for the **ForEach** command is:

ForEach (Var;{Expression1;Expression2;...Expressionx})

When running, the **ForEach** command evaluates the Expressions and inserts the result into the Var variable. The expressions can actually be simple numbers, text, calculations, and just about anything else that can be stored in a variable.

Additional codes you add between the **ForEach** and **EndFor** commands are repeated with each cycle of the loop.

You may wish to format the **ForEach** command within the macro as follows so you can readily identify each of the expressions.

Example

```
ForEach (Example;{
        2+2;
        2+4;
        2+6;
        2+8}
)
        NumStr (Example;0;Example)
        Type (Example)
        HardReturn()
EndFor
```

The **ForEach** loop is repeated a total of four times, one for each expression. On the first loop, the expression 2+2 is evaluated, and the result (4) is placed into the variable Example. The content of the variable is then converted to a string and printed in the document. A hard return follows each typed result to place it on a separate line. The loop then repeats. With each pass, another expression in the list is evaluated, until all expressions are complete. The final result is:

4
6
8
10

Fraction

The **Fraction** command assigns the fractional part of a number to a variable (the fractional part is the .34 in 12.34). The syntax for **Fraction** is:

Fraction (Var;Expression)

Var is the variable that holds the fractional value (starting with a zero and decimal point). Expression is a number, variable, or math expression.

Example

```
Fraction (FracVar;123.45)
     //   0.45 is assigned to FracVar

Fraction (FracVar; 3/2)
     //   0.5 is assigned to FracVar

Fraction (FracVar; 12)
     //   0.0 is assigned to FracVar

FracNumber:=3.1416
Fraction (FracVar;FracNumber)
     //   0.1416 is assigned to FracVar
```

The **Fraction** command is useful for returning just the remainder of a division expression (such as 3/2), and for returning just the cents portion of a dollar-and-cents numeric total.

GetNumber

GetNumber displays a pop-up message box, containing a title bar, prompt, input box, and two command buttons (OK and Cancel). The **GetNumber** box accepts only valid numeric input.

Input	Result
1234	Accepted
878762.28	Accepted
$192.98	Not accepted (not a number)
-9287	Accepted

Following is the syntax for the **GetNumber** command:

GetNumber (Var;Message;Title)

- Var is the variable that contains the users response.
- Message is the prompt message displayed in the **GetNumber** box. This argument is a string. Enclose it in quotes, or you can use a string variable.
- Title is the text that appears in the title bar (caption) of the **GetNumber** box.

Here are some examples of using the **GetNumber** command:

Example

```
GetNumber (NumVar;"Enter a number from 1 to 10";"This is a test")
     //   Using literal strings for message and title

Message:="Enter a number from 1 to 10"
Title:="This is a test"
GetNumber (NumVar;Message;Title)
     //   Using string variables for message and title
```

In both cases the user's response is stored in the NumVar variable. This variable contains a numeric value.

GetString

What GetNumber does for numeric entry, **GetString** does for string entry. The syntax is almost the same, except for an additional optional argument:

GetString (StrVar;LENGTH=val;Message;Title)

The **LENGTH** keyword limits the input to the specified number of characters. The valid range for **LENGT**H is 1 to 256.

Example

```
GetString (StrVar;"Enter your name;"This is a test")      //   Without
LENGTH argument

Title:="This is a test"
GetString (StrVar;LENGTH=12;"Enter your name (no more than 12
     chars)";Title)
     //   With LENGTH argument
```

In both examples, the user's response is contained in StrVar. This variable contains a string value.

GetUnits

The third type of message entry box is **GetUnits**. It is designed to accept only valid measure values. It accepts numbers only, plus an optional units of measure suffix:

GetUnits (UnitVar;Message;Title)

Units of measure	**Suffix**
Inches	i or "
Centimeters	c
Points	p
WordPerfect Units (1/1200-inch)	w

You can use upper or lower case for the units of measure suffix.

Example

```
GetUnits (UnitVar;"Enter left margin;"This is a test")
```

The user's response is contained in UnitVar. The value contained in the variable is a unique variable type that can only be applied to WPWin product function commands that expect a units of measure value.

Go

Use **Go** to branch to the specified label inside the current macro. You can go to labels that are in front of or behind the **Go** statement. The **Go** command uses one argument, the name of the label to go to:

Go (Label@)

Example

```
    Go (Example@)
         ...
    Label (Example@)
         //   Goes to the label identified as Example@
```

Note that the label name used with **Go** should be 15 bytes or less (WPWin uses only the first 15 bytes of a label name to test for uniqueness), and that the label must end with a @ symbol.

If

The **If** command is used to build an **If** expression.

- **If** the expression is TRUE, the macro performs the commands following the **If** statement (or if no commands follow the **If** statement, the macro jumps to the end of the **EndIf** command).
- **If** the expression is FALSE, the macro performs the commands after the **Else** command, if any.

The syntax for **If** is:

If (Expression)

The result of the **If** expression is always either TRUE or FALSE, and is always used with the EndIf command. The Else command is also used with **If**, but is optional.

Example

```
If (ExampleVar = 10)
     Go (Start@)
Else
     Quit
EndIf
     //   Go to Start@ if TRUE; otherwise terminate macro

If (ExampleVar = 10)
Else
     Quit
EndIf
Go (Start@)
     //   Do the same as above
```

Both examples do the same thing. If the value in variable ExampleVar is equal to 10 (TRUE), go to the label named Start@. If the value in variable ExampleVar is not equal to 10 (FALSE), stop the macro.

Note that the equal sign is not the only operator you can use with the **If** statement, although it is the most common. You can also test for not equal to, greater than, less than, and others.

Integer

The Integer command assigns the integral (whole) part of a number to a variable (the integral part is the 12 in 12.34). The syntax for Integer is:

Integer (Var;Expression)

Var is the variable that holds the integral value (no trailing decimal point). Expression is a number, variable, or math expression.

Example

```
Integer (IntVar;123.45)
     //   123 is assigned to IntVar

Integer (IntVar; 3/2)
     //   1 is assigned to IntVar

Integer (FracVar; 0.12)
     //   0 is assigned to IntVar

IntNumber:=3.1416
Integer (IntVar;IntNumber)
     //   3 is assigned to IntVar
```

The **Integer** command is useful for returning just the whole part of a division expression (such as 3/2), and for returning just the dollars portion of a dollar-and-cents numeric total.

Label

The **Label** command identifies a specific area of the macro, and is used with the Go, Call, Case, OnCancel, OnError, and OnNotFound commands commands. The label itself is really inconsequential; what's important are the macro codes that follows, which together comprise a routine located through the use of Label.

The syntax for **Label** is:

Label (LabelName@)

The label name should be 15 bytes or less, as WPWin in uses only the first 15 bytes of a label name to test for uniqueness. The label name must always end with a @ symbol. The names can contain letter and number characters only, but the name cannot start with a number.

Example

```
Label (Example@)
Type ("This is the text for the Example@ label routine")
     //   Identifies the label as Example@, and prints
          the text that follows
```

Keep the following in mind when using the **Label** command:

- Each abel name can be used only once in a macro.
- To avoid typing errors and to decrease the size of your macros, however, try to limit label names to less than 8-10 characters.
- You should not use labels simply as "markers" in a macro. If you declare a label, you should also include a reference to it, using a **Go**, **Call**, **Case**, or **On**... command. WPWin will catch this as a syntax error.

Menu

The **Menu** command displays a "floating" pop-up menu anywhere on the screen. The menu is "expandable" so that it shows only as many elements as you have defined. WPWin allows for up to 26 menu items, shown in two columns. optional placement arguments allow you to position the menu anywhere on the screen.

Following is the syntax for the **Menu** command:

Menu (MenuVar;MnemonicType;HorzPos;VertPos;{Option1;...Optionx})

- MenuVar is the variable that contains the user's choice after it has beed selected from the menu.
- MnemonicType is either **DIGIT** or **LETTER**, depending on the type of quick-access character desired. If **DIGIT**, the menu items are shown with a number from 1 to 9 (maximum number of menu items: 9). If **LETTER**, the menu items are shown with a letter from A to Z (maximum number of menu items: 26). **DIGIT** and **LETTER** are keywords, not strings, so don't enclose them in quotes.
- HorzPos is the horizontal location, in pixels, of the upper-left corner of the menu (relative to the upper-left corner of the WPWin window). Omit this argument to have WPWin automatically center the menu horizontally on the screen.
- VertPos is the vertical location, in pixels, of the upper-left corner of the menu (relative to the upper-left corner of the WPWin window). Omit this argument to have WPWin automatically center the menu vertically on the screen.
- Option1...Optionx are the text for one or more menu items. These Option parameters are strings, so be sure to enclose them in quotes, or use string variables.

Note the special syntax, including the braces ("curly brackets") that enclose the menu options. Be sure to include all syntactical elements of the menu, or WPWin will display a error message.

In the following examples the **Menu** command is shown with formatting as a visual aid to help you provide for all arguments. You are free to format the **Menu** command in any way you desire.
`

Example

```
Menu (MenuVar;
    DIGIT;
        ;;{
              "Option 1";
            "Option 2";
              "Option 3"}
)
 //     DIGIT mnemonics, three options;
          centered menu (no VertPos and HorzPos)

Menu (MenuVar;
    LETTER;
    100;100;
            "Menu line 1";
              "Menu line 2";
            "Menu line 3";
              "Last menu line"}
)
/   LETTER mnemonics, four options;
```

```
menu placed at 100 pixels horizontally and vertically
from the upper left corner of the WPWin window
```

Determining The Selection

The **Menu** command only displays a pop-up menu on the screen. Select an item, and the number of the menu item (starting from 1 and going down the list) is stored in the MenuVar variable. You can then use additional commands, such as an **If** or **Case** expression, to test for the value in MenuVar, and take appropriate action. Here's an example:

Example

```
Menu (MenuVar;
        DIGIT;
       ;;{
                          "Write a new letter";
             "Save this letter";
           "Print this letter"}
)
Case (MenuVar;{
       0;Cancel@;
       1;WriteNew@;
       2;Save@;
    3;Print@};
       Default@
)
 ...
Label (Cancel@)
    <cancel macro because Cancel key was pressed at menu>
Label (WriteNew@)
    <create a new letter here>
Label (Save@)
    <save the current letter here>
Label (Print@)
    <print the current letter here>
```

Notice that the **Menu** will return the value of 0 (zero) if the Cancel key was pressed when the menu was displayed. You can ignore cancels by not providing for a match in the **Case** or **If** expression (leave out the test for 0), or by turning cancel trapping off with **CancelOff**. When using a **Case** structure, you can test for the 0 condition by inference by leaving out a case match for 0, and letting the macro fall through to the Default@ label.

The **DIGIT** and **LETTER** keywords do not affect the return value that's stored in the MenuVar variable. The return value is always a number, and is never any more than the number of items in the menu.

NewDefault

Use NewDefault to specify a new product prefix default. The syntax for **NewDefault** is:

NewDefault (ProductPrefix)

ProductPrefix is a unique user-specified two letter code that associates a product prefix to a product identifier, as set earlier using the Application command. For example,

Example

```
Application (wp;wpwp)
```

associates the wp product prefix with the wpwp product ID. You can now use the **NewDefault** command to make wp the default product prefix.

Note that you can use the **Application** command or the **NewDefault** command to make the wp product prefix the default. Without a default specified, all product function commands (like HardReturn, Type, and Tab) would have to include the appropriate product prefix, as in

Example

```
wp.HardReturn()
wp.Tab()
```

and so forth. The product prefix (in this case wp) identifies these function commands as belonging to the wpwp product ID. Setting a default frees you from having to specify which product function commands go to which application:

Example

```
Application (wp;wpwp;Default)
     //   Specifies wp as the default product prefix

Application (wp;wpwp)
NewDefault (wp)
     //   NewDefault specifies wp as the default product prefix
```

Use **NewDefault** to change the default product prefix starting from a specific spot in the macro onward. The **NewDefault** command must physically occur before any associated product commands are encountered in the macro, and after the **Application** command that associates the product prefix with a product ID. During compiling, WPWin searches the macro from top to bottom looking for **NewDefault** commands. It ignores the flow of the macro, as set by **Go** and **Call** commands, or located within an **If** structure.

NumStr

NumStr stands for Number-to-String. Use this command to convert a numeric value to a a string. The NumStr command uses three arguments:

NumStr (StrVal;DecimalPlaces;NumVal)

- StrVal is a variable and is created by the **NumStr** command. It contains the string of the converted numeric value.
- DecimalPlaces is the number of digits to the right of the decimal point you want included in the string. Leave out this argument and WPWin will automatically provide six digits to the right of the decimal point. Enter 0 to exclude the fractional portion of the number. Valid values for DecimalPlaces is from 0 to 16.
- NumVal is a numeric value. It can be an literal value (entered as a number right in the command); more often it will be a variable.

You'll most often use **NumStr** to convert a numeric value to a string to you can **Type** it, or include it in a **MacroStatusPrompt** or other message (WPWin cannot directly **Type** or display numeric values).

Example

```
NumStr (StrVal;0;15)
     //   "15" assigned to StrVal

NumVal:=100029
NumStr (StrVal;;NumVal)
     //   "100029" assigned to StrVal

NumVal:=3.14159265
NumStr (StrVal;0;NumVal)
     //   "3" assigned to StrVal

NumVal:=3.14159265
NumStr (StrVal;2;NumVal)
     //   "3.14" assigned to StrVal

NumVal:=3.14159265
NumStr (StrVal;5;NumVal)
     //   "3.14159" assigned to StrVal

NumVal:=3.14159265
NumStr (StrVal;;NumVal)
     //   "3.141592" assigned to StrVal
```

OnCancel

OnError

OnNotFound

The **OnCancel**, **OnError**, and **OnNotFound** commands are called "exception handlers," because they handle exceptions that may occur during macro execution. The commands "trap" cancel, error, and not found conditions, respectively. All three are optional. If the macro does not contain an On... exception handler command, WPWin will terminate the macro as a safety precaution. The On... commands let you devise macros that perform some other action other than quitting when a cancel, error, or not found condition occurs.

OnCancel

The **OnCancel** command indicates what should happen if a macro encounters a cancel condition. Cancel conditions are generated by one or more of the following:

- Cancel key is pressed
- Cancel button is clicked in a dialog or message box
- Macro encounters a cancel from within itself or in a nested macro.

The syntax for **OnCancel** (and the other two exception handlers for that matter) is:

OnCancel (Label@)

or

OnCancel Call (Label@)

Label@ is a label provided elsewhere in the macro. There can be any number of commands following the label.

- **OnCancel** (Label@) sends the macro to the specified label. The commands that follow this label are responsible for controlling the macro flow. The macro can either end, branch off to some other routine, or just about anything else.
- **OnCancel Call** (Label@) temporarily branches the macro to the specified label. When the labeled subroutine is complete, a Return command sends the macro back to the point immediately after wherever the macro was before the cancel occurred (note: the macro does not necessarily return to the point right after the **OnCancel Call** command).

Some examples:

Example

```
OnCancel (Cancel@)
     ...
     <rest of macro>
Label (Cancel@)
```

```
     <do something here>
     Quit
     //   On cancel condition, macro branches to Cancel@, where
          it performs some action (defined by you), then quits
 OnCancel Call (Cancel@)
     ...
     <rest of macro>
Label (Cancel@)
     <do something here>
     Return
     //   On cancel condition, macro calls Cancel@ subroutine,
          where it performs some action, then returns to the
          point immediately following the last command executed
          before the cancel
```

OnError

OnError tells WordPerfect what to do if an error is detected during macro execution. The error can occur in the macro, in WordPerfect, or in DOS. The syntax and use is the same as with **OnCancel**.

OnNotFound

OnNotFound tells Tells WordPerfect what to do if a search ends with a *not found* message. The syntax and use is the same as **OnCancel**.

Pause

The **Pause** command temporarily pauses macro execution. Choose the Macro-Pause command to restart the macro. The Pause command is used without an argument.

Example

```
MacroStatusPrompt (On!;
     "Enter your name, then choose Macro-Pause to go on")
Pause
     ...
```

The **Pause** command has limited use. You may find more flexibility in the PauseKey product function command. The **PauseKey** command is among those product function commands that are often used in macros.

Prompt

Use the **Prompt** command to display a message box. The box contains:

- Your message (up to 512 characters).
- A title (caption) in the title bar.
- An OK button to dismiss the box.
- A Cancel button dismiss the box and cancel the macro (the effect of the Cancel will depend on your use of the **CancelOff** and **OnCancel** commands).

The Prompt command requires five arguments:

Prompt (Title;Message;Icon;HorzPos;VertPos)

- The Title is the text that appears in the title bar of the **Prompt** box. This is a string, so enclose it in quotes or use a string variable.
- The Message is the actual message displayed in the box. You can enter up to 512 characters. WPWin automatically wraps words to fit neatly on each line. You cannot enter your own line breaks. This is a string, so enclose it in quotes, or use a string variable.
- The Icon argument allows you to specify an icon to include in the box, if any. See the table below for valid values. You can also omit this parameter and WPWin will display the Prompt box without an icon. However, you must still add the proper number of semi- colons or or a syntax error will result.
- HorzPos is the horizontal location, in pixels, of the upper-left corner of the **Prompt** box (relative to the upper-left corner of the WPWin window). Omit this argument to have WPWin automatically center the box horizontally on the screen.
- VertPos is the vertical location, in pixels, of the upper-left corner of the **Prompt** box (relative to the upper-left corner of the WPWin window). Omit this argument to have WPWin automatically center the box vertically on the screen.

Icon Type	**Icon parameter value**
No icon	Blank or 0
STOP icon	1
QUESTION icon	2
EXCLAMATION icon	3
INFORMATION icon	4

Example

```
Prompt ("Title";"This is the message";;;)
     //   Title and message only; no parameters for Icon or
          position

Prompt ("Title";"This is the message";1;;)
     //   Same as above, but with STOP icon

Prompt ("Title";"This is the message";3;250;100)
     //   Title and message text; EXCLAMATION icon;
          HorzPos of 250 pixels; VertPos of 100 pixels
```

Dismissing the Prompt with EndPrompt

The **Prompt** box will appear on the screen either until the macro ends, or the macro encounters an EndPrompt command. A popular use of the **Prompt** command is to display a message for a particular amount of time, then remove the box. The following example shows how this is done, as well as how to use variables for the Title and Message strings.

Example

```
Title:="Prompt Example"
Message:="Read this message.  It will remain for two seconds."
Prompt (Title;Message;0;;;)
Wait (20)
EndPrompt
```

You can also use the **PauseKey** product function command to wait for a particular keypress before the macro continues. The following example displays a message, then waits for you to press the **[Enter]** key, or press the OK button in the **Prompt** box.

Example

```
Title:="Prompt Example"
Message:="Read this message, then press Enter or click OK."
Prompt (Title;Message;0;;;)
PauseKey(Enter!)
EndPrompt
```

Quit

The **Quit** command terminates all macro execution. The command is used without an argument.

Repeat

The Repeat command begins a controlled **Repeat**/**Until** loop. Refer to the Until command for more information.

Return

The **Return** command is used with all **Call** commands (Call, Case Callcmd_Case, OnCancel Call, etc.). It marks the end of a subroutine. When **Return** is encountered, the macro jumps back to the point immediately after the **Call**ing command. The **Return** command is used without an argument.

Return also finds use in other situations as well:

- When **Return** is placed outside of a subroutine, it terminates the macro.
- When **Return** is placed outside of a subroutine, and after a **Chain** command, it terminates the current macro, and immediately plays the macro specified by the **Chain** command.
- If the **Return** command is placed outside a subroutine, and the macro has been nested, WPWin ends the current macro, and returns to the original (nesting) macro.

ReturnCancel

ReturnError

ReturnNotFound

The **ReturnCancel**, **ReturnError**, and **ReturnNotFound** commands perform the same basic function as the **Return** command, but in addition to returning control to the previous routine or macro, these trigger a cancel, error, or not found condition. All three commands are used without an argument.

The Return... commands are beneficial when used in conjunction with the OnError, OnCancel, and OnNotFound exception handling command.

Run

The **Run** commands nests to named macro. After the nested macro finishes, WPWin returns to the original macro, and continues execution where it left off. The **Run** command accepts one argument: the name of the macro to run:

Run ("macroname")

Note that the name of the macro is enclosed in quotes.

Example

```
Run ("DOLIST.WCM")
     //   Nests to the macro DOLIST.WCM

Assign (PlayMacro;"c:\wpwin\macros\dolist.wcm")
Run (PlayMacro)
     //   Nests to the macro in the PlayMacro variable (note path provided
with macro name)
```

WPWin insists that the macro be compiled before it will **Run** it. You need to play the macro at least one if it is not yet compiled (you have never used it before, for example). Or, you can use the Macro Facility program to compile the macro without actually playing it.

Finding Macros

To ensure that your macros are always found, include an explicit directory path so WPWin is sure to locate them. This works fine for macros that are used on a specific system, where you know the directory structure, but it isn't an acceptable approach if the macro may be used on other computers, which may have different directory paths.

In these instances, you can use the GetWPData product function command to return the directory used for storing macro files (as specified in the Location of Files dialog box).

Example

```
GetWPData (MacroPath;MacroPath!)
Run (MacroPath + "mymacro.wcm")
```

GetWPData returns the full path (with disk drive) of the macro directory. The MacroPath variable is a string.

Speed

The **Speed** command slows down macro execution by the indicated amount. **Speed** is most often used as a "debugging" tool to analyze the progression of a macro. You can also it, for example, to slow down macro execution so that some process is graphically shown to the user.

The **Speed** command requires one argument; the amount of time to wait between executing each command. The time is expressed in tenths of a second.
Speed (TenthsOfSecond)

Example

```
Speed (10)
     //   Wait one second between commands

Speed (1)
     //   Wait one-tenth of a second between commands

Speed (17)
     //   Wait 1.7 seconds between commands

Speed (0)
     //   No waiting between commands
```

Valid values for the TenthsOfSecond parameter are any whole numbers between 0 (no waiting) and 600 (60 seconds).

You can use as many **Speed** commands in a macro as you wish. Execution time will increase or decrease as the macro encounters the additional **Speed** commands.

StrLen

The **StrLen** command counts the number of characters in a string (typed explicitly in the **StrLen** command) or in a string variable. You might use **StrLen** to limit text entry, to determine if a variable contains more characters that it should, or to develop sophisticated macros that "parse" the contents of variables.

Following is the syntax for StrLen:

StrLen (LenVar;StringVar)

LenVar is the return value of the **StrLen** command. It contains the length of the StringVar string or variable. Here are some examples.

Example

```
StrLen (LenVar;"This is a test")
     //   LenVar contains 14

StringVar:="This is also a test"
StrLen (LenVar;StringVar)
     //   LenVar contains 19
```

Remember that the return value is a number, so if you want to Type the number into the document, you have to convert the number to a string first (use NumStr).

Example

```
Assign (MyString;"This is a test")
StrLen (NumBytes;MyString)
NumStr (sNumBytes;0;NumBytes)
Type (sNumBytes)
     //   Macro Types "14"
```

The **StrLen** command counts the number of characters in a string, which may or may not reflect the actual number of bytes in the string. Use the ByteLen command to count the number of bytes in a string.

StrNum

The **StrNum** command stands for String-to-Number. Use **StrNum** to convert a number string to a numeric value. The number string should contain only digits, a decimal point, and a minus sign. If WPWin encounters any alphabetic characters, it will ignore the remainder of the string. **StrNum** requires two arguments:

StrNum (NumVal;StrVal)

NumVal is the return variable of the **StrNum** command. It contains the converted StrVal string. StrVal is a string, so it should be either a string enclosed in quotes or a string variable.

Example

```
StrNum (NumVal;"12345")
     //   NumVal contains the number 12345

StrVal:="12345"
StrNum (NumVal;StrVal)
     //   NumVal contains the number 12345

StrVal:=12345A6789"
StrNum (NumVal;StrVal)
     //   NumVal contains the number 12345 (additional characters
          ignored after encountering A)
```

StrPos

StrPos checks the referenced string for a particular character or group of characters (called a substring).

- If the substring is found in the referenced string: The return value is the starting position, in number of characters, of the substring.
- If the substring is not found in the referenced string. The return value is zero (0).

Following is the syntax for the **StrPos** command:

StrPos (ReturnVal;SubString;RefString)

RefString is the string to check. SubString is the string to find in RefString. ReturnVal is the starting position, in characters, of SubString (or 0, if not found).

Example

```
RefString:="This is a test"
SubString:="is"
StrPos (WhereIsIt;SubString;RefString)
     //   WhereIsIt returns 6, as "is" starts at the sixth
          character in the RefString variable
```

The **StrPos** command counts characters, not bytes. Extended WordPerfect characters are each four bytes (instead of one byte each, as with regular characters). See the ByteLen and BytePos commands for more information on how WPWin handles extended characters when counting by bytes.

SubByte

The **SubByte** command is used to extract a portion of the contents of a string (typically a variable). The resulting extracted portion is called a substring.

The command can be used to return the entire contents of the string, but in most cases, you'll use it to return only one or two characters. The **SubByte** command requires four arguments.

SubByte (ExtractString;StartPos;Length;OrigString)

- ExtractString is the return variable of the **SubByte** command, and contains the substring result. This is a string variable so it can be directly **Type**d into WPWin.
- StartPos is the number of bytes you want to start counting from (also called the offset). Byte 1 is the beginning of the string.
- Length is the number of bytes from the string you want to extract, beginning at StartPos.
- OrigString is the original string you want to use. The contents of ExtractString will be taken from OrigString. This is a string, so enclose it in quotes or use a string variable.

Example

```
Assign (OrigString; "This is my string.")
SubByte (ExtractString;1;4;OrigString)
Type (ExtractString)
     //   Types This

Assign (OrigString; "This is my string.")
SubByte (ExtractString;7;4;OrigString)
Type (ExtractString)
     //   Types s my
```

The **SubByte** command returns only as much of the original string (OrigString) as it can if you specify a length that extends beyond the actual length of the string. You'll get nothing if you specify a start point that is beyond the actual length of the original string.

Note that the StartPos and Length parameters of the **SubByte** command are bytes, not characters. All of the characters in the standard IBM character set (letters, numbers, symbols, Extended IBM Characters) use one byte each. So, a string such as "This is a test" returns a value of 14 when used with ByteLen, as "This is a test" contains 14 one-byte characters.

The remaining 1,200+ of WPWin's special extended characters are composed of four bytes each. See ByteLen for an explanation of the format of extended characters. You need to use care when using the **SubByte** command with extended characters, as you may extract only portions of characters, instead of a complete character.

Also available is the SubStr command, which extracts a string based on character count, not byte count.

SubStr

The **SubStr** command is used to extract a portion of the contents of a string (typically a variable). The resulting extracted portion is called a substring.

The command can be used to return the entire contents of the string, but in most cases, you'll use it to return only one or two characters. The **SubStr** command requires four arguments.
SubStr (ExtractString;StartPos;Length;OrigString)

- ExtractString is the return variable of the SubStr command, and contains the substring result. This is a string variable so it can be directly **Type**d into WPWin.
- StartPos is the number of characters you want to start counting from (also called the offset). Character 1 is the beginning of the string.
- Length is the number of characters from the string you want to extract, beginning at StartPos.
- OrigString is the original string you want to use. The contents of ExtractString will be taken from OrigString. This is a string, so enclose it in quotes or use a string variable.

Example

```
Assign (OrigString; "This is my string.")
SubStr (ExtractString;1;4;OrigString)
Type (ExtractString)
     //   Types This

Assign (OrigString; "This is my string.")
SubStr (ExtractString;7;4;OrigString)
Type (ExtractString)
     //   Types s my
```

The **SubStr** command returns only as much of the original string (OrigString) as it can if you specify a length that extends beyond the actual length of the string. You'll get nothing if you specify a start point that is beyond the actual length of the original string.

The **SubStr** command is often used in conjunction with a **For** loop to "parse" (or scan) a string into its individual characters. Examples of this technique can be found throughout this book, but here's a sample:

Example

```
CharCount:=0
OrigString:="WordPerfect for Windows"
StrLen (LenOfString;OrigString)
For (Count;1;Count<=LenOfString;Count+1)
     SubStr (TestChar;Count;1;OrigString)
     If (TestChar = "o")
          CharCount:=CharCount + 1
     EndIf
EndFor
NumStr (CharCount;;CharCount)
Type ("The letter 'o' appeared " + CharCount + " times.")
```

This macro counts the number of times the letter o appears in the string "WordPerfect for Windows" (the answer is 3, of course). Note the use of the **For** loop, and its Count variable, to specify a start position for the **SubStr** command. Each iteration of the loop the start position moves one character to the right, until the entire string has been parsed.

Until

The Until command is the "working" half of the **Repeat**/**Until** controlled loop structure.

- **Repeat** marks the beginning of the loop. Commands after Repeat are part of the loop.
- **Until** marks the end of the loop. Commands between **Repeat** and until are part of the loop. **Until** also provides the conditional expression that control the loop.

The expression used with **Until** is the same as that used in the If command. The result of the expression is always either TRUE or FALSE. As long as the expression evaluates to FALSE, the **Repeat**/**Until** loop continues. When the expression evaluates to TRUE, the loop is broken, and the macro continues. Note this unusual behavior of TRUE and FALSE conditions.

The syntax of the **Until** command is shown below, in context with the **Repeat** command.

Repeat
    <repeating commands>
Until (Expression)

Example

```
Count:=1
Repeat
     HardReturn()
     Count:=Count + 1
Until (Count >10)
```

This example enters 10 hard returns in the document, then stops. The **Until** expression is Count >10. Count starts at 1. With each iteration of the loop, Count is increased by 1. The first 10 times the loop is repeated Count >10 is FALSE, so the loop continues. On the 11th trip through, the expression is TRUE, and the loop is broken.

The **Repeat**/**Until** loop is what is known as an exit loop. The test expression isn't encountered until the end of the loop, so the loop must be completed at least one time, regardless of the outcome of the expression. Conversely, the While loop, described later in this chapter, is an entry loop. Its test expression appears at the beginning of the loop structure. The contents of a **While** loop may never be executed, depending on the outcome of the text expression.

Wait

The **Wait** command waits the specified period of time before continuing. Wait requires one argument: the amount of time to wait before continuing. The time is expressed in tenths of a second.

Wait (TenthsOfSecond)

Example

```
    Wait (10)
         //   Wait one second before continuing

    Wait (35)
         //   Wait 3.5 seconds before continuing
```

Valid values for the TenthsOfSecond parameter are any whole numbers between 0 (no waiting) and 600 (60 seconds).

WPWin accepts no input while the macro is **Wait**ing. You must wait through the **Wait** period until it is over.

While

The **While** command sets up a unique repeating loop that causes the macro to repeat a given set of instructions. The looping continues as long as the expression in the **While** statement is TRUE. When the While statement proves FALSE, the loop is broken and the macro continues. Commands between the **While** and **EndWhile** commands are considered part of the loop and are repeated.

The syntax of the **While** command is shown below, in context with the **EndWhile** command.

While (Expression)
    <repeating commands>
EndWhile

Example

```
Count:=1
While (Count <=10)
     HardReturn()
     Count:=Count + 1
EndWhile
```

This example enters 10 hard returns in the document, then stops. The **While** expression is Count <=10. Count starts at 1. With each iteration of the loop, Count is increased by 1. The first 10 times the loop is repeated Count <=10 is TRUE, so the loop continues. On the 11th trip through, the expression is FALSE, and the loop is broken.

The **While** loop is what is known as an entry loop. The test expression appears at the beginning of the loop structure. The contents of a **While** loop may never be executed, depending on the outcome of the text expression. Conversely, the Repeat/Until loop, described earlier in this chapter, is an exit loop. Its test test expression isn't encountered until the end of the loop, so the loop must be completed at least one time, regardless of the outcome of the expression.