µ §
A macro is a set of instructions that you can create for Microsoft Word to follow. You can use a macro to combine a series of actions into one step. Whenever you frequently repeat the same steps in Word, it's likely that creating a macro to perform the task can save you some time and effort.

You can use macros to configure and customize Word for many situations. You can add and delete menu items, move them around, and assign commands to key combinations. In some cases you may want to write special applications with Word. Using fields, macros, and templates you can create a document processing system that meets your specific needs.

The examples in this chapter highlight the programming possibilities available with Word. Although these examples are simple, they show some of the basics with which you can create your own programs with Word.

Writing a Macro

There are two ways to write a macro. You can create a macro by "recording" keystrokes you make. You can also write your macro by using the statements and functions of WordBASIC, the Word macro language.

For more information on specific statements and functions, see Macros: Reference.

Using Macro Record to Write a Macro

When you record a macro, you turn on the recorder and Word records all the actions you take until you turn off the recorder.

To record a macro:

1	Choose Macro Record (Alt,M,C). The Macro Record dialog box appears on the screen.
2	Type the name you want to give the macro, or accept the proposed name.
3	If you want, type a description of the macro in the Description box.
4	When you're ready to begin recording, choose OK. Word records any subsequent steps you perform. Once you start recording a macro, any keyboard or menu commands you choose are automatically recorded. The only mouse actions recorded are those that actually choose a menu command or dialog box item. For example, if you select text with the mouse, Word does not record that action.
5	To stop the macro recorder, choose Macro Stop Recorder (Alt,M,C).

The macro recorder is useful even if you don't want to record an entire macro. Recording all or part of a macro and then editing it is often faster than typing it from scratch, and you don't have to look up the syntax for every function and statement you want to use. You can also use the PauseRecorder and RecordNextCommand commands to help construct a macro. For more information on these statements, see Macros: Reference.

Using Macro Edit to Write a Macro

You can use the Macro Edit command to write your macro directly and then save it.

To write a macro directly:

1	Choose Macro Edit (Alt,M,E), type a name for the macro, and choose OK. The macro editing icon bar appears below the menu bar (see below).
2	Type the desired macro programming statements and functions.
3	Close the macro editing window (Alt,F,C).

Macro Editing Icon Bar

The macro editing icon bar includes a number of functions that can help you debug your macro programs. Press Alt+Shift+the underlined letter to choose an icon. The icons are described as follows:

Start/Continue:	Runs the active macro; changes from Start to Continue after a stop (such as after a Step).
Step:	Runs a single macro instruction and then stops. If the instruction is a subroutine, Word runs each instruction in that subroutine as a single step.
Step SUBs:	Runs a single macro instruction and then stops. If the instruction is a subroutine, Word runs that subroutine in its entirety as a single step.
Trace:	Runs the active macro, highlighting each instruction as it is carried out.
Vars:	Displays the variables the macro uses.
Global/Template	This text shows you the context (global or template) of the active macro. If the context is template, the template name is displayed.
(Name):	Displays the name of the active macro

Running a Macro

Note: Because an untested macro can create errors or alter your file, always make backup copies of your files before you test a new macro.

Once your macro has been written, you can run it by doing the following:

1. Choose Macro Run (Alt,M,R).
2. Type the name of the macro you want to run, or select a name from the list in the Run Macro Name box.
3. Choose OK.

A Sample Macro

The following procedures create a macro that automatically sets formatting properties for a document:

1. If you are in page view, switch to draft view or normal editing view.
2. If the status bar is not displayed, choose View Status Bar (Alt,V,S).
3. Choose Macro Record (Alt,M,C).
4. In the Record Macro Name box, type PageSetup to name the macro.

Note that the Global Context option is selected. This means the macro can be used in any document you open.

5. Choose OK. "REC" appears on the right side of the status bar, indicating that Word is now recording the keystrokes you make.

Now, choose the commands and options as you want them recorded:

1. Choose Format Document (Alt,T,D).
2. Change the default tab stops to 0.25".
3. Change the left margin to 1".
4. Change the right margin to 0.5".
5. Choose OK to accept these changes.
6. Choose Format Character (Alt,T,C).
7. Change the font to Tms Rmn.
8. Change the point size to 12.
9. Choose OK to accept these changes.
10. Choose Edit Header/Footer (Alt,E,H).
11. Select Footer and choose OK to open the header/footer pane.
12. Click the page icon to put the page number on the left side of the footer.
13. Press Tab twice, then click the date icon to put the date on the right side of the footer.
14. Choose Close.

To stop recording actions, choose Macro Stop Recorder (Alt,M,C). "REC" disappears from the status bar.

To test the new macro you have recorded:

1. Select a document.
2. Choose Macro Run (Alt,M,R).
3. Scroll through the list of macros and double-click PageSetup to start running the macro.

If you made any mistakes when recording the macro, messages will appear on the screen.

You can add the macro to the Macro menu for easier access. To add the PageSetup macro to the Macro menu:

1. Choose Macro Assign To Menu (Alt,M,M).
2. In the Assign Macro Name box, select PageSetup.
3. In the Menu box, select &Macro.
4. Choose Assign, and then choose Close.

The PageSetup macro now appears on the Macro menu. To run this macro, press Alt,M,P. If you want to remove the macro from the Macro menu, choose Macro Assign To Menu, select PageSetup, and choose Unassign.

You can use the Macro Edit command to look at the command list created when you record a macro. To look at the PageSetup macro:

1. Choose Macro Edit (Alt,M,E).
2. Type or select PageSetup and choose OK.

The PageSetup macro appears in the macro editing window. These are the statements and functions that comprise the PageSetup macro. They correspond to the actions you recorded.

Macro Programming Concepts

This section describes in more detail some of the features of WordBASIC, Word's macro language.

The WordBASIC Language

WordBASIC macros exist on three "layers," much like the three file levels DOS uses. If you are an experienced DOS user, you know that DOS executes files in this order: filename.exe, filename.com, and filename.bat. You can therefore have three files with the same file name-but with different extensions-and DOS runs them according to this convention.

WordBASIC's three layers are the template layer, the global layer, and the command menu layer. These layers are described in the following table:

Layer	**Description**
Template	Includes only those macros based on the specified template
Global	Includes macros you create that are available to all documents
Built-in command	Includes the commands on all the default Word menus and assigned to default key combinations

When you run a macro, Word searches for it in the following order: template layer, then global layer, then the built-in command layer. So, if you create a macro with the same name as a built-in macro, your version will be executed instead of the original version. If Word cannot find a macro, a message to that effect is displayed.

Remember the order of macro execution when naming macros. If you have a macro at the template layer and a macro at the global layer with the same name and you want the global macro executed, you must either rename one of the macros or precede the macro with the Super prefix statement. The Super prefix forces Word to ignore the current layer and start searching the next layer. For example, the following macro, called FormatDocument, disables mirror margins when the Format Document command is chosen:

```
Sub MAIN
        Dim dlgrec As FormatDocument
        GetCurValues dlgrec
Again:
        Dialog dlgrec
        If dlgrec.MirrorMargins = 1 Then
                Beep
                MsgBox "Mirror Margins have been disabled by this macro"
                dlgrec.MirrorMargins = 0
                Goto Again
        End If
        Super FormatDocument dlgrec
End Sub
```

Auto Macros

Word reserves special names for macros you can create to alter aspects of Word's behavior. These are called "auto macros." Word recognizes a macro whose name begins with "Auto" as a macro that runs automatically when the situation it applies to arises. You supply the actual steps for the auto macro.

You can prevent an auto macro from running by holding down the Shift key when you perform the action that triggers the macro.

AutoNew:	The AutoNew macro runs after you create a new document based on the current template.
AutoOpen:	The AutoOpen macro runs after you open a file with File Open, File Find, or the list of documents at the bottom of the File menu.
AutoExec:	The AutoExec macro runs when you start Word. This macro makes it easy to instruct Word to automatically make adjustments when you start it. You can prevent AutoExec from running by typing the /m switch when you start Word (winword /m).
AutoClose:	The AutoClose macro runs when you close a document (File Close, Document Control Close, File Exit, or closing Windows).
AutoExit:	The AutoExit macro runs when you quit Word.

WordBASIC Statements and Functions

WordBASIC includes both statements and functions. A statement performs an action, such as italicizing text. A function produces, or "returns," a number or a set of characters that represent information. Functions appear in the text with parentheses () following them.

WordBASIC includes three types of statements and functions: utility statements and functions, BASIC statements and functions, and dialog control definition statements. These statements and functions are described in more detail in Macros: Reference. The following table briefly describes each type:

Utility statements and functions	Miscellaneous statements and functions that allow you to get information needed by a macro. Includes dialog box equivalents, which are equivalent to Word commands that produce a dialog box. For example, the WordBASIC statement UtilRenumber is equivalent to choosing the Utilities Renumber command and displaying the resulting dialog box.
BASIC statements and functions	Statements and functions taken directly from the Microsoft QuickBASIC language.
Dialog control definition statements	Statements that create customized dialog boxes. For example, the GroupBox statement creates a box with a title grouping several options together in a dialog box.

WordBASIC is a subset of the BASIC programming language, similar to Microsoft QuickBASIC. One difference between WordBASIC and prior forms of BASIC is that the main program must be located inside a subroutine called MAIN. Sub MAIN is always the first line of a WordBASIC macro; End Sub is always the last line (see "Subs," later in this chapter). Nothing is allowed outside this subroutine except global variable declarations, such as Dim and Declare, and the other Sub and Function definitions. The following example shows a small program in both BASIC and WordBASIC:

```
BASIC                WordBASIC
Print "Hello!"       Sub MAIN
End                          Print "Hello!"
                     End Sub
```

The result of the first program in BASIC displays "Hello!" on the screen. In WordBASIC, "Hello!" appears in the status bar at the bottom of the screen.

In WordBASIC you can use a colon (:) to separate two statements or functions on the same line. You can use a backslash (\ ) at the end of a line of code to indicate that the code continues on the next line.

Data Types

WordBASIC supports two basic data types: strings and numbers. Word uses double-precision, floating-point numbers. Strings can contain up to 32,000 characters, depending on the amount of memory available. The following are examples of these data types.

```
String                                    Number
Text$ = "this is a string of characters"  Sales = 270
Print Text$                       Print Sales
```

Variables are usually local to the subroutine or function in which they are used. If your macro consists of several subroutines or functions and you want to make a variable globally available to subroutines and functions within the macro, declare them with a Dim statement located outside the Sub MAIN. If you want to permanently store variables, store them in a file or glossary.

String variables must have a trailing dollar sign; for example, Name$. Numeric variable names require no special character. Unlike standard BASIC, WordBASIC does not support integer variables. Word does support multidimensional arrays of strings or values. Array variables are declared with the Dim statement and can be redimensioned with the Redim statement.

The syntax for the Dim statement is as follows:

Syntax: Dim [Shared] Var [(Size)] [, Var [(Size)]...]

The Dim statement declares a variable's type and allocates storage space for the variable. If Shared is used, then the variable is global; if not, the variable is local to the Sub or Function. If the variable is global, the Dim statement must be located outside the Sub or Function. If the variable is local, the Dim statement must be located inside the Sub or Function. Dim can also be used to declare global scalar (nonarray) variables.
Arrays allow you to assign multiple values to a single variable. The macro can then determine which value to access, as shown in the following example:

Program listing	Effect
Sub MAIN	
Dim MonthSales(12)	Dimensions a one-dimensional array to hold 12 values
For Month = 1 To 12	Sets up a loop for the macro to cycle through 12 times
Input "Please enter the sales for this month", MonthSales(Month)	Ask the user for input; the value input is assigned to the array element called MonthSales(Month); Month will vary from 1 to 12 as the loop progresses
Next Month	Increments Month by 1; returns to the For statement; when the value reaches 12, the macro continues to the next line

End Sub

Using the array form shortens the program. Without an array, each month would have to be entered as an individual variable.

If a macro uses dialog boxes or commands that use dialog boxes, a third data type is available, the dialog record. A dialog record consists of a list of "fields." Each field in a dialog record contains the value of an element in the dialog box; the value is a number in some cases and a string in others. Some dialog record fields can accept either a number or a string; in these cases, Word converts a string such as "1 in" to the equivalent number of printer's points. This feature is only available for some dialog record fields. These fields are followed by a dollar sign enclosed in brackets ([$]) in the macro statement syntax in this chapter. This is a convention used for your information only. Do not include the [$] when you are writing dialog records in macros. You can set or read a specific field of a dialog record by specifying the field name, preceded by a period (.).
Dim can be used to dimension dialog records. The syntax follows:

Dim DialogRecord As DialogBox

In the above syntax Dim allocates to DialogRecord the storage space and associated field types for DialogBox.
To copy the current elements of a dialog box to a dialog record, use the GetCurValues statement (see Macros: Reference for more information on the GetCurValues statement).

The Dialog statement can be used to display a dialog box with the values taken from the specified dialog record (see Macros: Reference for more information on the Dialog statement).

Program listing	**Effect**
Sub MAIN	
Dim dlg As FormatDocument	Creates a dialog record with empty fields
GetCurValues dlg	Places the current values of the Format Document command into the record
If dlg.MirrorMargins = 0 Then	Toggles the mirror margins field of the record dlg.MirrorMargins = 1
Else dlg.MirrorMargins = 0	
Dialog dlg	Displays the dialog box
FormatDocument dlg	Performs the action using the values specified in the dialog record
End Sub	

Talking With The User

WordBASIC differs from other versions of BASIC in the features it provides for communications with the user. The familiar Print and Input statements are available, but with some limitations. More powerful and useful are the MsgBox (message box) and InputBox statements, which allow WordBASIC programs to use dialog boxes, buttons, and icons as easily as other BASIC programs use Print and Input.

For more complex interaction, WordBASIC lets you use any of Word's built-in dialog boxes or even create your own using check boxes, buttons, list boxes, and other features.

Print

The biggest difference between the WordBASIC Print statement and that of other versions of BASIC lies in where it places text or numbers on the screen. Print operates only in the single-line status bar at the bottom of the Word window. If you have not turned on the View Status Bar option, running a Print statement creates a special one-line bar at the bottom of the window. The text displayed by the Print statement remains on the screen until the next action that causes Word to update the status bar. At that time, if you have turned on the View Status Bar option, Word replaces the text with the updated status information. If you have not turned on View Status Bar, the special Print status bar disappears.

Here is an example:

```
SUB Main
        Print "Greetings. This is a test"
End Sub
```

WordBASIC does not support special forms of Print, such as Print@, Print Tab, and Print Using. You can use Print to display string or numeric variables, string or numeric constants, and expressions in any combination.

The following are all valid uses of Print:

```
Print Name$
Print Total
Print "Hello"
Print 365.25
Print Abs(WordCount / 3)
Print "Cost is ";Item + Tax;" as supplied by: ";Vendor$
Print FirstCount,SecondCount,ThirdCount
```

When you use commas as delimiters between multiple items in a Print statement, the next printed item starts at the next print tab position. When you use semicolons as delimiters between items, the next item starts immediately following the previous one.

Unlike in some BASICs, in WordBASIC ending a Print statement with a trailing semicolon does not cause the output of the next Print statement to appear on the same line. Instead, the next Print statement clears the status bar and overwrites the output of the previous Print statement.

Even with these new limitations, Print remains an effective command. It is useful when a program must update information on a frequent basis or when the information must be presented in an accessible but unobtrusive way.

MsgBox
In many cases, you will need your programs to display a message that you don't want the user to overlook. For example, you may want the user to verify something before proceeding with an irreversible change to a document. This is the sort of task for which the MsgBox statement was created.

Message boxes have four elements:

A Message.	A string that appears in the message box.
A Title.	A string that appears at the top of the message box. This is optional. If no title is provided, Microsoft Word is used.
Buttons.	At a minimum, a message box has an OK button. Other possible buttons are Cancel, Abort, Retry, Ignore, Yes, and No.
An icon.	The default is no icon, but you can display one of four icons to reinforce the purpose of the message string.
	In its simplest form, a MsgBox statement consists of the command and a message string, as in these examples:

```
MsgBox "Macro complete"
MsgBox Note$
```

In each of these cases, the title of the message box is "Microsoft Word", and the box has an OK button and no icon. Adding a title string is simple:

```
MsgBox "Macro complete", "Salutation 1"
```

A numeric Type argument controls which icon and buttons are displayed, and which button is the default. Here is an example:

```
MsgBox "OK to reformat?", "Make Two Column", 292
```

This statement creates a message box with a message of "OK to reformat", a title of "Make Two Column", Yes and No buttons with No as the default, and a question-mark icon.

Input
Usage of the Input statement is much the same in WordBASIC as in other versions of BASIC, except that prompts created by WordBASIC's Input appear only in the status bar. You can use Input to prompt the user to provide values for variables. These variables can be string, numeric, or a combination of the two.

For example, to prompt the user for a value for a single variable, use Input statements like these:

```
Input Length
Input Address$
```

To prompt for a list of variables, use statements like these:

```
Input Height, Width
Input UserID$, Password$
```

To provide values for a list of variables, the user must respond with the appropriate values seperated by commas, as in these examples:

```
6,3
JamesK, Tribble
```

No matter how many variables you use in an Input statement, Input always attempts to divide the user's response at a comma. If you want to allow responses that contain commas, use a Line Input statement. Line Input works exactly the same as Input with one important difference; you can specify only a single string variable. Here's an example:

Line Input Address$

If Word encounters a comma in the response, it returns the comma as part of the string. With some imagination and the use of WordBASIC's string functions, you can use Line Input to accept almost any kind of value—numeric or string. An example of creative Line Input usage can be found later in this section, in the discussion of InputBox$().

To request a value for a variable by using a customized prompt, use a statement like this one:

```
Input "What point size of type for headline";Size
```

Note that you don't have to include a question mark in the prompt string, because the prompt created by Input is a question mark. The prompt displayed by this statement is

```
What point size of type for headline?
```

If you include a question mark in the prompt string, as in

```
Input "What point size of type for headline?";size
```

the prompt users see in the status bar is

```
What point size of type for headline??
InputBox$()
```

Although the unobtrusive way in which Print places text in the status bar is sometimes desirable, you may want a prompt that halts the program until the user supplies input to draw more attention to itself. The prompt created by an Input statement can easily be overlooked, but there is no missing the result of the InputBox$() function. Even if Word is minimized to an icon while the macro is running, the InputBox$() prompt appears in the middle of the screen, demanding attention.

Even though MsgBox is a statement and InputBox$() is a function, their syntaxes are similar. Like MsgBox, InputBox$() takes three arguments. The first two arguments are strings, the first of which is the prompt string and the second the title string. The title string is again optional. Examples of valid usage are

```
File$ = InputBox$("Name of the file to load?")
Search$ = InputBox$("Type search string:", "Search for Last")
MsgBox InputBox$("What is your message?")
```

The last of these examples may seem strange, but remember that InputBox$() is a function and you can use it anywhere you might use a string expression. In this case, Word displays the input box first to prompt the user for a string. The string that the user enters then appears in the message box.

The following examples are also valid:

```
Print InputBox$("String to print?","Print")
Sort$ = Left$(InputBox$("Sort key (first four letters used)"),4)
```

Although the InputBox$() function can return only a string, you can continue it with WordBASIC's Val() function to input a numeric variable, as in the following example:

```
Number = Val(InputBox$("Input a number from 1 to 10"))
```

The Val() function returns the numeric value, if any, of a string. If in response to the prompt created by the above statement the user typed 10, then Val() would assign the value 10 to Number. If, however, the user typed Ten or Microsoft, Val() would assign the value 0 to Number, because Ten and Microsoft contain no digits, (the only things Val() can detect).

You use the InputBox$() function's optional third argument to insert a default string into the text area. For example:

```
Name$ = InputBox$("Find what name?", "File Search", "No name selected")
```

Here, "No name selected" appears in the text area as the default. The user can replace the default, edit it, or accept it as is.

Expressions

Word can evaluate complex numeric and string expressions. An expression is any valid combination of variables, numbers or strings, and functions that evaluate down to a single result. This result can be a number, a string, or, in the case of logical expressions, a True or False condition. (In WordBASIC, all logical expressions return –1 if True and 0 (zero) if False.)

At its very simplest, an expression is only one of the above items. The following are examples of simple expressions:

```
Heading$
LineCount
80486
"Word"
Rnd()
```

More often, an expression consists of several elements in combination, as in the following:

```
Page1Lines + Page2Lines
"The " + CarName$ + "automobile is the best seller this week"
(2 * Pi) * Radius
(Rain Or Snow) And (Hail Or Sleet)
Int((365 - Days) / 7)
FirstQtr + SecondQtr + ThirdQtr + FourthQtr > LastYear
```

You can use an expression anywhere that a constant or a variable can be used, as in these examples:

```
Print Left$(FirstName$,1) + ". " + LastName$
LineDown ParaLong + 1, Switch
If Profit > 0 and Month$ = "Jan" Then Print "Good Year!"
```

Numeric Expessions

Bitwise operators (Not, And, and Or) convert numbers to 16-bit integers and then process the individual bits of the number in binary format.

Not of –1 is False. Not of any other number, including 0 (zero), is True. Therefore, be careful when using bitwise operators with non-Boolean functions.

When Word evaluates numeric expressions, it performs multiplication and division before addition and subtraction. To have Word perform operations in a different order, use parentheses, as shown in the second example that follows:

14 * 5 - 6	Multiplies 14 by 5 and then subtracts 6 from the result
14 * (5 - 6)	Subtracts 6 from 5 and then multiplies 14 by the result

The following table lists the numeric and logical operators.

Operator	Description
–	Negates the number
*	Multiplies the numbers
/	Divides the first number by the second number
Mod	Rounds the numbers to integers, divides the first number by the second number, and returns the remainder
+	Adds the numbers
–	Subtracts the second number from the first number
=	Tests for equality
<>	Tests for inequality
>	Tests whether the first number is greater than the second number
<	Tests whether the first number is less than the second number
<=	Tests whether the first number is less than or equal to the second number
>=	Tests whether the first number is greater than or equal to the second number
Not	Performs the bitwise complement of the number; also toggles the state of the expression between True and False
And	Performs a comparison that returns True if both of the conditions are met
Or	Performs a comparison that returns True if either of the conditions are met

String Expressions

If you compare strings using logical operators, Word first converts the strings to ASCII values and then uses the resulting values in the comparison. For example, when Word evaluates the expression

If "Apple" < "Orange" Then Print "Apple is less than Orange."

Word first converts the relational expression into a value that represents True or False. It then evaluates the If statement using that value and performs the print operation accordingly.

String Functions

WordBASIC includes most of the string-manipulation functions available in other versions of BASIC. With these functions, you can create a string using ANSI character values, disassemble an existing string into its component parts, or shuffle the characters in a string like cards in a deck.

The following table lists WordBASIC's string functions.

Example	Result	Description
Asc("ABC")	65	Returns the ANSI code for the first character in the specified string
Chr$(67)	C	Returns a one-character string based on the numeric ANSI code specified
Instr("Television","v")	5	Returns a number representing the location of a specified search string in a target string
LCase$("HELLO")	hello	Converts a string into lowercase characters
Left$("Automobile",4)	Auto	Returns a specified number of characters from the left of a target string
Len("Elephant")	8	Returns a numeric value indicating the length in characters of a target string
Mid$("ABCDEFGH",4,3)	DEF	Returns a specified number of characters from a target string starting at a specified index point
Right$("Microsoft",4)	soft	Returns a specified number of characters from the right of a target string
Str$(10)	"10"	Returns a string representation of the specified numeric value
String$("*",3)	***	Returns the first character of a target string repeated a specified number of times
UCase$("hello")	HELLO	Converts a string into uppercase letters
Val("1234")	1234	Returns the numeric value of the target string
		Some of WordBASIC's string functions duplicate or approximate Word command equivalents. You can duplicate other command equivalents by using a combination of WordBASIC commands and string and other functions.

The best approach to use for text-manipulation tasks is dictated by the nature of the individual task. In general, command equivalents are usually faster, more compact and, when it comes to manipulating document text, easier to program; however, string functions offer more flexibility.

Beyond this, the major difference between the use of string functions and the use of Word command equivalents is that the command equivalents work on text in an open document, and string functions work on string variables. However, WordBASIC provides ways to work around this fundamental incompatibility.

Converting Document Text to String Variables

You can use the Selection$() function to load text from an open Word document into a string variable. Selection$() loads either the currently selected text or, if no text is selected, the first character following the insertion point. You must work within the standard limitations on WordBASIC string variables when using Selection$(). You cannot load more than 32,000 characters into one string variable, and you cannot store more than a total of 64,000 characters in all the variables in a program. If you select more than 32,000 characters, only the first 32,000 characters are loaded into the variable and an error is generated. All formatting is stripped from the text when it is placed in the variable. Selection$() does not delete the selected text from the document or change its formatting. In this respect, it is much like the Edit Copy command. The syntax for Selection$() is as follows:

Text$ = Selection$()

The following are all valid uses of Selection$():

```
Cut$ = Selection$()
Big$ = Selection$() + OldText$
Print Selection$()
First$ = Left$(Selection$(),1)
```

Converting String Variables to Document Text

You use the Insert command to insert strings from a WordBASIC program into a Word document at the insertion point. If document text is selected when Insert is run, Word inserts the string before the selection.

After the Insert operation, Word moves the insertion point to the end of the inserted text. Word assigns the current style attributes to the inserted text.

The syntax for Insert is as follows:

Insert Text$

The following are all valid uses of Insert:

```
Insert Name$
Insert "To our valued customers:"
Insert "Dr. "+ Name$
Insert Chr$(9)
```

The last example illustrates how the Chr$() function can be used with Insert to insert nonprinting characters into a document. (Chr$(9) is a tab character.)

Control Structures

The following control structures can be used to program Word macros. Their actions are similar to those used in Microsoft QuickBASIC.

For...Next

```
Syntax:   For CounterVariable = Start To End [Step Increment]
          Statement(s)
          Next [CounterVariable]
```

Executes the statements between For and Next as many times as it takes the CounterVariable to go from the Start value to the End value. The Increment is the value to increment the counter (usually 1).

For names the CounterVariable and specifies the Start and End values in the range of the CounterVariable. These values can be expressed as constants, as variables derived before the start of the loop, or as expressions that compute a range of values for the CounterVariable.

The Increment can be a positive or a negative number; positive numbers increase the count, negative numbers decrease the count. If Increment is omitted, the default is 1. An example follows:

```
For Month = 1 To 12
        Print Month
Next Month
```

Goto

Syntax: Goto Label/LineNumber

Branches unconditionally to an optional label or line number. The syntax for labels and line numbers follows:

```
Label:      [Statement]
LineNumber Statement
```

If...ElseIf...Else...End If

```
Syntax:   If Condition Then Statement(s) [Else Statement(s)]
Syntax:   If Condition1 Then
          Statement(s)
          [ElseIf Condition2 Then
          Statement(s)]
          [Else
          Statement(s)]
          End If
```

Performs conditional execution or branching, depending on the expressions. The conditions in an If...ElseIf...Else...End If block can be any numeric expressions in WordBASIC.

WordBASIC evaluates the conditions in the order in which they appear and executes the statements corresponding to the first condition resulting in a True (nonzero) value.

If tests for a specified condition. If the condition exists, the operations following the Then statement are executed. Else is performed if none of the If or ElseIf conditions evaluate to True. Else is optional; if it is not included and all previous conditions are False, Word takes no action.

To build conditional expressions, use the relational operators (=, <>, <, >, >=, <=) and the Boolean operators (And, Or, Not).

An example of the If...Then control structure follows:

```
If Sales > 300 Then Print "Sales were more than 300"
```

An example of the If...ElseIf...Else...End If control structure follows:

```
If Sales > 300 Then
        Print "Sales were more than 300!"
        BestEver = Sales
ElseIf Sales > 280 Then
        Print "Sales were OK"
Else
        Print "Another year goes by"
End If
```

On Error
Syntax: On Error Goto Label
Syntax: On Error Resume Next
Syntax: On Error Goto 0

Normally, when WordBASIC encounters an error in a program, a message explaining the error is displayed and the program is terminated. The On Error control structure allows the programmer to "trap" an error so that the program can perform its own error handling. The first form of the control structure, On Error Goto, causes the program to branch to the specified label whenever an error occurs. At the end of the error handling, it is necessary to reset the variable Err to 0 for further errors to be trapped.

The second form of the control structure, On Error Resume Next, causes all errors to be ignored.

The third form of the control structure, On Error Goto 0, disables the error trapping. Once an error has been trapped, the special variable Err contains the code for the error that occurred. For more information on error codes, see the full Technical Reference.

Be careful when using error trapping. The statement causing the error may have performed some, but not all, of its action, thereby causing other statements that rely on that action to fail.

Select Case
Syntax:
```
        Select Case Expression
        Case CaseExpression
        Statement(s)
        [Case Else
        Statement(s)]
        End Select
```

This control structure is similar to a multi-line If statement in that a statement or group of statements is executed based on the result of some expression. With Select Case, however, only one expression is evaluated, even though there may be several groups of statements.

The Expression is evaluated and the result is compared with the CaseExpression. A CaseExpression is preceded by the keyword Case and may be followed by a single value, a list of values separated by commas, a range of values separated by the keyword To, or a relation started with the keyword Is, followed by a relational operator (=, <>, <, >, <=, or >=) and an expression.

The Expression is compared with all the values given in each CaseExpression until a match is found. If a match is found, the Statement(s) following the CaseExpression are executed. If there is no match and there is a Case Else, those Statement(s) are executed.

An example follows:

```
Select Case Int(Rnd() * 10) - 5
Case 1,3
        Print "one or three"
Case Is > 3
        Print "Greater than three"
Case -5 to 0
        Print "Between -5 and 0 (inclusive)"
Case Else
        Print "Must be 2"
End Select
```

Stop
Syntax: Stop
Stops a running macro and displays a message that the macro was interrupted.

While...Wend

Syntax:
```
While Condition
    Statement(s)
    Wend
```

Repeats the statements in the block while the Condition is True. If the Condition is initially False, the loop is never executed.

A While...Wend loop uses a conditional expression to determine the number of times the Statement(s) are executed. WordBASIC evaluates the Condition each time the Statement(s) are executed. As long as the Condition evaluates as True, the Statement(s) are executed. When the Condition evaluates as False, the Statement(s) are no longer executed.

A conditional statement is any numeric expression. To build conditional expressions, use the relational operators (=, <>, <, >, >=, <=) and the Boolean operators (And, Or, Not). The evaluation results are -1 for True and 0 for False.

An example follows:

```
Sub MAIN
Count = 0
StartOfDocument
EditSearch "macro"
While EditSearchFound()
    Count = Count + 1
    EditSearch "macro"
Wend
Print "macro was found ";Count; " times"
End Sub
```

Subs

A "subroutine" is a group of statements that performs a task. In WordBASIC, a macro begins with a Sub MAIN statement and ends with an End Sub statement. Every macro in Word must be set up as a subroutine with these statements. You cannot nest subroutines; that is, a subroutine cannot be located within another subroutine. The syntax follows:

Syntax:
```
Sub Name [ParameterList]
    Statement(s)
    End Sub
```

The simplest macros consist of only one subroutine. As macros get more complicated, they are usually written in smaller, separate units. If your macro performs the same action in different parts of the program, you can write another subroutine. Suppose you want the computer to beep before each message is displayed. One way to do this is shown in the following listing:

```
Sub MAIN
    BeepMsg "Are you sure you want to quit?", 0
    BeepMsg "Don't you want to save your work first?", 0
    BeepMsg "This is your last chance. Choose OK to quit.", 65
End Sub

Sub BeepMsg (msg$, type)
    Beep
    MsgBox msg$,, type
End Sub
```

User-Defined Functions

You can define new functions in a manner similar to subroutines. Instead of using the keyword Sub, you use the Function keyword. The syntax follows:

Syntax:
```
Function Name [ParameterList]
    Statement(s)
    End Function
```

Defines a function. The ParameterList is a list of variables, separated by commas, for receiving arguments to the function. Functions without parameters should not have parentheses. The statements are used to produce a value that the function returns when called. An example follows:

```
Function RndInt(n)
    RndInt = Int(Rnd()*n)
End Function
```

The Rnd() function returns a fractional value between 0 and 1. Sometimes it is useful to generate an integral random number between 0 and some specific value; the preceding example does this. The Function RndInt(n) line tells Word that a new function is being defined and that it takes a single numeric parameter called n. The second line indicates that the value of the function is the formula Int(Rnd()*n).

Every user-defined function includes an implied variable with the same name as the function. Assigning a value to that variable defines the value that is to be returned from the function. A function can contain more statements above and/or below the assignment, just as if it were a subroutine.

A user-defined function returns a numeric value unless the name is terminated with a dollar sign ($), which indicates that the function returns a string.

File Input-Output

WordBASIC supports the standard BASIC stream input-output (I/O) statements and functions. However, record-based file I/O is not supported.

You can have up to four files open at one time. Each file is assigned a number from 1 to 4. This identifies the file to Word's macro processor. The # symbol indicates that the expression following it is a file number. For example, Open "RBOW.TXT" For Input As #1 opens the specified file for input and assigns the file number 1 to it. When accessed with other file statements, the number 1 indicates which of the open files to use.
The following macro searches in a text file for a given string (case sensitive):

```
Sub MAIN
Rem Sets up a dialog record
Dim dlgrec As FileOpen
Rem Fills the dialog record with the defaults
GetCurValues dlgrec
dlgrec.Name = "*.TXT"
Rem Allows the user to change the values
On Error Goto bye
Dialog dlgrec
On Error Goto 0
Search$ = InputBox$("Search for what string?")
Rem Connects the specific file to stream 1
Open dlgrec.Name For Input As #1
Print "Searching"
Rem While not at the end of file 1 Reads one line of the file into Text$
Rem and exits from the search loop otherwise loops again
While Not (Eof(1))
        Line Input #1, Text$
        If Instr(Text$, Search$) Then
                MsgBox Search$ + " was found in file: " + dlgrec.Name
                Goto Found
        End If
Wend
Beep
Rem Beeps at end of file
MsgBox Search$ + " was not found in file: " + dlgrec.Name
Found:
Rem Closes the file
Close
bye:
End Sub
```

Special Bookmarks

In addition to user-defined bookmarks, there are several reserved bookmarks that can be used with macro statements such as CmpBookmarks, GetBookmark$(), and EditGoTo.

Bookmark	**Definition**
\Sel	Current selection
\PrevSel1	Previous selection 1 where editing occurred (nil at start)
\PrevSel2	Previous selection 2 where editing occurred (nil at start)
\StartOfSel	Start of selection
\EndOfSel	End of selection
\Line	Current line (first of selection)
\Char	Current character (first of selection)
\Para	Current paragraph (first of selection)
\Section	Current section (first of selection)
\Doc	Entire document
\Page	Current page
\StartOfDoc	Beginning of document
\EndOfDoc	End of document
\Cell	Cell
\Table	Table
\HeadingLevel	A heading level

Macros: Reference

This chapter is a reference for constructing macros. It contains the syntax and a description of each of the functions and statements in WordBASIC. These statements and functions are divided into the following sections:

Utility statements and functions
Dialog control definition statments

Introduction

The WordBASIC language consists of statements and functions described in the following sections. A statement performs an action; Bold 1, for example, makes the selection bold. A function produces, or "returns," a number or string of characters that represents information. Most functions do not perform any action, but some do.

Those that do perform an action usually return a value indicating the success or failure of that action. A function is always followed by parentheses. For example, Overtype 1 is a statement; Overtype() is a function. If a function ends with $, it returns a string of characters. For example, the StyleName$() function returns a string of characters representing the style name of the selection.

For Boolean operators, if a function returns 0 (zero), False is implied. Any other value implies True. If a function can only be True or False, -1 is returned for True.

All statements that insert text are affected by the state of the Typing Replaces Selection option of the Utilities Customize command. If this option is turned on, inserted text overwrites selected text.

Measurements for statements should be entered in points (1/72 inch).

Macro programs have access to system information such as free memory and software version numbers. Be aware, however, that free memory changes constantly. Values returned may be only an approximation of free memory.

Statements can take arguments. In this chapter, a dollar sign ($) follows arguments that accept a string of characters. Some arguments take a value or a string. The string can be Auto, in the case of certain measurements, or a string such as 1 in, 2 cm, and so on. Word converts these measurements to points. In this chapter, these arguments are followed by a dollar sign in brackets ([$]). This is a convention adopted for your information only; do not use the dollar sign when you supply the arguments for actual macros.

Dialog Box Equivalents

Some statements are dialog box equivalents. That is, each of the statement arguments is equivalent to an option in the dialog box for a corresponding command on the command menu. The following conventions are used:

Check box equivalents take the following values: 1 checks the box; 0 (zero) unchecks the box; -1 indicates that the status of the box is unknown.

Option button groups can only have one option selected in the group. Option button equivalents take the following values: 0 selects the first option in the group; 1 selects the second option in the group, and so on; -1 indicates that the status is unknown or ambiguous.

List box equivalents vary. Some take a string value and others follow the conventions used for option buttons.
Combo boxes are similar to list boxes, but you can either select from the list or type the desired response. Combo box equivalents take string values.

You can set up the arguments to dialog box equivalent statements in two ways: you can use the positional form by listing the argument values after the statement keyword, separated by commas; or you can use the keyword form by following the statement keyword with the argument name, preceded by a period and followed by an equal sign (=), which is in turn followed by the argument value. An example of each method follows:

```
Positional form:    EditSearch "Cost of Goods", 1
Keyword form:       EditSearch .Search = "Cost of Goods", .WholeWord = 1
```

The order in which argument values are specified is important when using the positional form, so this form is most useful for statements with relatively few arguments. When using the keyword form, you need to include only those arguments that you want to change from the default, so this form is best if you want to avoid looking up or memorizing the syntax for statements with numerous arguments. You can use both forms for one statement, but arguments specified in the keyword form must follow arguments specified in the positional form.

Command buttons carry out actions. Command button equivalents are not arguments in the traditional sense and are not included in statement syntax. They are discussed in the text following the syntax line. Command button equivalents can be specified only with the keyword form and must be appended to the argument list.

You can specify only one command button per statement.

Dialog Control Definition Statements
You can create your own dialog boxes and customized menus with Word macros. The control statements used in dialog box construction are described in this section. For more information on dialog box construction, see the full Technical Reference.

In the syntax lines, the following arguments are used:

Argument	Meaning
x	Horizontal position of the item in 1/8 system font character width units
y	Vertical position of the item in 1/12 system font character width units
dx	Width of the item in 1/4 system font character width units
dy	Height of the item in 1/8 system font character width units

Begin Dialog
Syntax: Begin Dialog UserDialog [x, y,] dx, dy
Starts the dialog box declaration. The dx and dy arguments are the width and height of the dialog box (relative to the given x and y coordinates). If x and y are not supplied, then the dialog box is positioned automatically by Word at the point where dialog boxes usually appear on the screen.

CheckBox
Syntax: CheckBox x, y, dx, dy, Text$, .Field
Creates a check box. When the dialog box is used .Field contains the current setting: if the value is 0, the box is not checked; any other value means the box is checked. The result is a numeric field with the value 0 (zero) (not checked) or 1 (checked) or -1 (grayed) in the dialog record returned from Dialog.

ComboBox
Syntax: ComboBox x, y, dx, dy, Array_Variable$, .Field
Creates an expanded combo box with the list box filled from the Array_Variable$. When the dialog box is used, .field contains the current setting, your selected string, returned from Dialog.

Dialog
Syntax: Dialog DialogRecord
Displays the dialog box specified by DialogRecord, for editing. After editing, you can store edits in DialogRecord by choosing OK or lose edits by choosing Cancel. Choosing Cancel produces a run-time error that you can trap with On Error.

End Dialog
Syntax: End Dialog
Ends the definition of the dialog box.

GroupBox
Syntax: GroupBox x, y, dx, dy, Text$
Creates a box with a title. A GroupBox does not have a result.

ListBox
Syntax: ListBox x, y, dx, dy, Array_Variable$, .Field
Creates a list box control filled with the strings in Array_Variable$. When the dialog box is used, .Field contains the current setting, the index of your selected choice, returned from Dialog.

OKButton and CancelButton
Syntax: OKButton x, y, dx, dy
Syntax: CancelButton x, y, dx, dy
If you choose the OK button, the macro continues. If you choose the Cancel button, an error is generated. This error can be trapped with On Error. For more information on On Error, see Macros: Introduction.

OptionGroup and OptionButton
Syntax: OptionGroup .Field
Syntax: OptionButton x, y, dx, dy, Text$
OptionGroup begins the definition of a series of related option buttons. Within the group only one button may be active (on) at a time. The .Field argument is set to a value between 0 (zero) and n, which represents the value of the currently active button.

Text
Syntax: Text x, y, dx, dy, Text$
Creates a box of static text. Text does not have a result. Text statement must precede the dialog box control it is associated with.

TextBox
Syntax: TextBox x, y, dx, dy, .Field
Creates an edit control.

WordBasic Programming Language

The following is an alphabetic list of all WordBasic statements and functions. For statements and functions organized according to the purpose they serve, see WordBasic Cross Reference.

Abs
Activate
ActivateObject
AllCaps
AppActivate
AppInfo$()
AppMaximize
AppMinimize
AppMove
AppRestore
AppSize
Asc()
Beep
Begin Dialog...End Dialog
Bold
BookmarkName$()
Call
Cancel
CancelButton
CenterPara
ChangeCase
ChangeRulerMode
CharColor
CharLeft
CharRight
ChDir
CheckBox
Chr$()
Close
ClosePane
CloseUpPara
CmpBookmarks()
ColumnSelect
ComboBox
CommandValid()
ControlRun
CopyBookmark
CopyFile
CopyFormat
CopyText
CountBookmarks()
CountFiles()
CountFonts()
CountFoundFiles()
CountGlossaries()
CountKeys()
CountLanguages()
CountMacros()
CountMenuItems()
CountMergeFields()
CountStyles()
CountWindows()
Date$()
DDEExecute
DDEInitiate()
DDEPoke
DDERequest$()
DDETerminate
DDETerminateAll
Declare
DeleteBackWord
DeleteWord
Dialog
Dim
DisableAutoMacros
DisableInput
DocClose
DocMaximize
DocMove
DocRestore
DocSize
DocSplit
DocumentStatistics
DoFieldClick
DoubleUnderline
EditClear
EditCopy
EditCut
EditFind
EditFindChar
EditFindClearFormatting
EditFindFound()
EditFindPara
EditFindStyle
EditFootnoteContNotice
EditFootnoteContSep
EditFootnoteSep
EditGlossary
EditGoTo
EditLinks
EditObject
EditPaste
EditPasteSpecial
EditPic
EditRepeat
EditReplace
EditReplaceChar
EditReplaceClearFormatting
EditReplacePara
EditReplaceStyle
EditSelectAll
EditUndo
EmptyBookmark()
EndOfColumn
EndOfDocument
EndOfLine
EndOfRow
EndOfWindow
Eof()
Err
Error
ExistingBookmark()
ExpandGlossary
ExtendMode()
ExtendSelection
File1
File2
File3
File4
FileClose
FileCreateDataFile
FileCreateHeaderFile
FileEditDataFile
FileExit
FileFind
FileName$()
FileNew
FileNewDefault
FileOpen
FileOpenDataFile
FileOpenHeaderFile
FilePrint
FilePrintDefault
FilePrintMerge
FilePrintMergeCheck
FilePrintMergeReset
FilePrintMergeSelection
FilePrintMergeSetup
FilePrintMergeToDoc
FilePrintMergeToPrinter
FilePrintPreview
FilePrintPreviewMargins
FilePrintPreviewPages
FilePrintSetup
Files$()
FileSave
FileSaveAll
FileSaveAs
FileSummaryInfo
FileTemplate
Font
FontSize
FootnoteOptions
FormatBorder
FormatCharacter
FormatColumns
FormatDefineStyleBorder
FormatDefineStyleChar
FormatDefineStyleFrame
FormatDefineStyleLang
FormatDefineStylePara
FormatDefineStyleTabs
FormatFrame
FormatLanguage
FormatPageNumber
FormatPageSetup
FormatParagraph
FormatPicture
FormatSectionLayout
FormatStyle
FormatTabs
For...Next
FoundFileName$()
Function...End Function
GetBookmark$()
GetCurValues
GetGlossary$()
GetProfileString$()
GetToolButton()
GetToolMacro$()
GlossaryName$()
GoBack
Goto
GroupBox
GrowFont
HangingIndent
Help
HelpAbout
HelpActiveWindow
HelpContext
HelpIndex
HelpKeyboard
HelpTutorialGstart
HelpTutorialLword
HelpUsingHelp
HelpWPHelp
Hidden
HLine
HPage
HScroll
IconBarMode
If...ElseIf...Else...End If
Indent
Input
Input$()
InputBox$()
Insert
InsertAnnotation
InsertBookmark
InsertBreak
InsertChart
InsertColumnBreak
InsertDateField
InsertDateTime
InsertDrawing
InsertField

InsertFieldChars
InsertFile
InsertFootnote
InsertFrame
InsertIndex
InsertIndexEntry
InsertMergeField
InsertObject
InsertPageBreak
InsertPageField
InsertPageNumbers
InsertPara
InsertPicture
InsertSymbol
InsertTableOfContents
InsertTimeField
InStr()
Int()
IsDirty()
IsExecuteOnly
Italic
JustifyPara
KeyCode()
KeyMacro$()
Kill
Language
LCase$()
Left$()
LeftPara
Len()
Let
Line Input
LineDown
LineUp
ListBox
LockFields
Lof()
MacroCopy
MacroDesc$()
MacroName$()
MenuMacro$()
MenuMode
MenuText$()
MergeFieldName$()
Mid$()
MkDir
MoveText
MsgBox
MsgBox()
Name...As
NextCell
NextField
NextObject
NextPage
NextTab()
NextWindow
NormalStyle
OK
OKButton
On Error
OnTime
OpenUpPara
Open...For...As
OptionButton
OptionGroup
OtherPane
OutlineCollapse
OutlineDemote
OutlineExpand
OutlineLevel()
OutlineMoveDown
OutlineMoveUp
OutlinePromote
OutlineShowFirstLine
Overtype
PageDown
PageUp
ParaDown
ParaUp
PauseRecorder
PrevCell
PrevField
PrevObject
PrevPage
PrevTab()
PrevWindow
Print
PushButton
Read
RecordNextCommand
Redim
Rem
RemoveFrames
RenameMenu
RepeatFind
ResetChar
ResetFootnoteContNotice
ResetFootnoteContSep
ResetFootnoteSep
ResetPara
Right$()
RightPara
RmDir
Rnd()
RulerMode
SaveTemplate
Seek
Select Case...Case Else...End Select
SelectCurAlignment
SelectCurColor
SelectCurFont
SelectCurIndent
SelectCurSpacing
SelectCurTabs
Selection$()
SelInfo()
SelType
SendKeys
SentLeft
SentRight
SetDirty
SetEndOfBookmark
SetGlossary
SetProfileString
SetStartOfBookmark
Sgn()
Shell
ShowAll
ShowAllHeadings
ShowHeading1
ShowHeading2
ShowHeading3
ShowHeading4
ShowHeading5
ShowHeading6
ShowHeading7
ShowHeading8
ShowHeading9
ShowVars
ShrinkFont
ShrinkSelection
SmallCaps
SpacePara1
SpacePara2
SpacePara15
Spike
StartOfColumn
StartOfDocument
StartOfLine
StartOfRow
StartOfWindow
Stop
Str$()
Strikeout
String$()
StyleName$()
SubScript
Sub...End Sub
Super
SuperScript
TabLeader$()
TableColumnWidth
TableDeleteCells
TableDeleteColumn
TableDeleteRow
TableGridlines
TableInsertCells
TableInsertColumn
TableInsertRow
TableInsertTable
TableMergeCells
TableRowHeight
TableSelectColumn
TableSelectRow
TableSelectTable
TableSplit
TableSplitCells
TableToText
TabType()
Text
TextBox
TextToTable
Time$()
ToogleFieldDisplay
TogglePortrait
ToggleScribbleMode
ToolsBulletListDefault
ToolsBulletsNumbers
ToolsCalculate
ToolsCompareVersions
ToolsCreateEnvelope
ToolsGetSpelling
ToolsGetSynonyms
ToolsGrammar
ToolsHyphenation
ToolsMacro
ToolsNumberListDefault
ToolsOptions
ToolsOptionsGeneral
ToolsOptionsGrammar
ToolsOptionsKeyboard
ToolsOptionsMenus
ToolsOptionsPrint
ToolsOptionsSave
ToolsOptionsSpelling
ToolsOptionsToolbar
ToolsOptionsUserInfo
ToolsOptionsView
ToolsOptionsWinini
ToolsRecordMacro
ToolsRepaginateNow
ToolsRevisionMarks
ToolsSorting
ToolsSpelling
ToolsSpellSelection
ToolsThesaurus
UCase$()
Underline
UnHang
UnIndent
UnLinkFields
UnLockFields
UnSpike
UpdateFields
UpdateSource
Val()
ViewAnnotations
ViewDraft
ViewFieldCodes
ViewFootnotes
ViewHeaderFooter
ViewHeaderFooterLink
ViewMenus()
ViewNormal

ViewOutline
ViewPage
ViewRibbon
ViewRuler
ViewStatusBar
ViewToolbar
ViewZoom
ViewZoom100
ViewZoomPageWidth
ViewZoomWholePage
VLine
VPage
VScroll
While...Wend
Window()
Window1
Window2
Window3
Window4
Window5
Window6
Window7
Window8
Window9
WindowArrangeAll
WindowMainDoc
WindowName$()
WindowNewWindow
WindowPane()
WordLeft
WordRight
WordUnderline
Write

Abs()

n = Abs(n)
Returns the absolute unsigned value of n.

Example
Print Abs(5)
Print Abs(-5)
Both instructions display the value 5.

_____
See also
Standard Basic Statements and Functions

Activate

Activate WindowText$ [,PaneNum]
Activates the specified window.

WindowText$	The name of the window to activate, as it appears in the title bar, or the full path of the document.

PaneNum	**The number of the pane to activate:**
1 or 2	Top pane
3 or 4	Bottom pane

Example
Document$ = "DOC5.DOC"
Pane = 1
Activate Document$, Pane
Activates the upper pane of the window titled DOC5.DOC.

_____
See also
Window Statements and Functions

ActivateObject

ActivateObject
Equivalent to double-clicking the selected embedded object.

_____
See also
View Statements and Functions

AllCaps

AllCaps [On]
n = AllCaps()
Adds or removes the All Caps character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format; mixed and related formats, such as Small Caps, also return -1).
1	If all of the selection is in the format.

_____
See also
Formatting Statements and Functions

AppActivate

AppActivate WindowText$ [,Immediate]
Activates the specified window in an application other than Word. For Word windows, use Activate.

WindowText$	The name of the application window to activate, as it appears in the title bar.

Immediate	**Specifies when to switch to the other application:**
0	If Word does not have the focus, Word flashes its title bar, waits for the user to give the focus to Word, and then activates the application.
1	Word immediately switches the focus to the other application, even if Word does not have the focus.

Examples

The instructions
AppActivate "Microsoft Excel", 1
and
Program$ = "Microsoft Excel"
Immed = 1
AppActivate Program$,Immed
activate the window named Microsoft Excel.

Note	Be careful when using this command, because window names can change. Many applications append the name of the working file to the application name used in the window title bar or when the application is maximized. For example, the Windows Cardfile application might have window names such as: Cardfile - (untitled) Cardfile - PHONE.CRD

_____

See also
Dynamic Data Exchange (DDE) Statements and Functions

AppInfo$()

a$ = AppInfo$(TypeOfInfo)
Returns information about the state of the Word application.

TypeOfInfo	**A numeric code specifying the type of information to return:**
1	The environment string; for example, "Windows 3.0".
2	The version number of Word; for example, "2.0".
3	Returns -1 if Word is in a special mode; for example, CopyText or MoveText mode.
4	X position of the Word window, measured in points from the left of the screen. (Returns -3 when maximized, due to the window's borders.)
5	Y position of the Word window, measured in points from the top of the screen. (Returns -3 when maximized, due to the window's borders.)
6	Width of the active document workspace, in points.
7	Height of the active document workspace, in points.
8	Returns -1 if the application is maximized, and 0 if the application is restored.
9	Total conventional memory.
10	Amount of conventional memory available.
11	Total expanded memory.
12	Amount of expanded memory available.
13	Returns -1 if a math coprocessor is installed, 0 if not.
14	Returns -1 if a mouse is present, 0 if not.
15	Amount of disk space available.

Examples

ver$ = AppInfo$(2)
MsgBox ver$, "Microsoft Word Version", 64
Displays a message box containing the version number of Word.
Values are returned as strings.
Use Val(AppInfo$()) to convert the string to a number, if appropriate:
DiskFree = Val(AppInfo$(15))
Print DiskFree
Displays the available disk space as a numeric value instead of as string data.

Note	Several AppInfo$() arguments deal with memory. These values have limited use in Windows, because the available memory can change from moment to moment, due to swapping of the Word program code and based on whether or not documents have been saved.

_____

See also
Environment Statements and Functions

AppMaximize

AppMaximize
n = AppMaximize()
If the Word window is less than full screen, zooms the window to full screen size. If the window is already full screen, performs the same action as AppRestore. The function form returns a nonzero value if the window is maximized.

Example
If AppMaximize() = 0 Then AppMaximize

_____
See also
Window Statements and Functions

AppMinimize

AppMinimize
n = AppMinimize()
If Word is not minimized, minimizes the Word window to an icon. If Word is already minimized, performs the same action as AppRestore. The function form returns a nonzero value if the window is minimized.

Note If an untrapped error occurs in a macro while Word is minimized, the macro halts, Word remains minimized, and the Word icon flashes. When Word is maximized, an error message box that indicates the nature of the error is displayed.

_____
See also
Window Statements and Functions

AppMove

AppMove XPos, YPos
Moves the Word window to the specified position, relative to the upper-left corner of the screen. Word cannot perform this action if the application is maximized.

XPos, YPos The coordinates of the upper-left corner of the Word workspace, in pixels.

Example
AppMove 20, 40
Moves the Word window 20 pixels right and 40 pixels down from the upper-left corner of the screen.

_____
See also
Window Statements and Functions

AppRestore

AppRestore
logical = AppRestore()
Restores the Word window from a maximized/minimized state. The function form returns a nonzero value if the Word window is not in the restored state.

_____
See also
Window Statements and Functions

AppSize

AppSize Width, Height
Resizes the Word window. This statement is available only if the window is in the restored state. Word cannot perform this action if the application is maximized.

Width The width of the Word window, in pixels.
Height The height of the Word window, in pixels.

Examples
AppMove 0, 0
AppSize 480, 350
Places the Word window at the upper-left corner of the screen and extends it to the lower-right corner of a standard EGA screen. The location of the lower-right corner varies depending on the type of video display used. Use AppMaximize to make the window full-screen on any type of display.
AppSize 240, 175

AppMove 120, 87
This example sizes the Word window to half-screen size and places it in the center of a standard EGA screen.

_____
See also
Window Statements and Functions

Asc()

n = Asc(A$)
Returns the character code of the first character in A$. Asc is short for the ASCII character set used with earlier versions of BASIC and is so named for compatibility purposes. The actual codes returned are those used by Microsoft Windows -- ANSI.

Example
Print Asc("Signature authority")
Displays the value 83, corresponding to the code for the character S.

_____
See also
Standard Basic Statements and Functions

Beep

Beep [BeepType]
Causes the computer's speaker to beep. A typical use of Beep is for signaling the end of a long process.

BeepType	The type of beep: 1 (or omitted), 2, 3, or 4. This value is passed to the operating environment, but current operating environments do not make use of the value. BeepType is provided for compatibility with future versions of Microsoft Windows.

_____
See also
Standard Basic Statements and Functions

Begin Dialog...End Dialog

Begin Dialog UserDialog [x,] [y,] dx, dy [, Title$]
...dialog definition instructions
End Dialog
Begins and ends a dialog box declaration record.

x, y	The coordinates of the upper-left corner of the dialog box, in increments of 1/8th (for x) and 1/12th (for y) of the system font. If not supplied,Word centers the dialog box on the screen.
dx, dy	The width and height of the dialog box, in increments of 1/8th (for dx) and 1/12th (for dy) of the system font.
Title$	The caption to use in the title bar of the dialog box; the default is "Microsoft Word." For more information, see "Using Macros" in the Microsoft Word User's Guide.

_____
See also
Dialog Box Statements and Functions

Bold

Bold [On]
n = Bold()
Adds or removes the bold character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format.
1	If all of the selection is in the format.

_____
See also
Formatting Statements and Functions

BookmarkName$()

a$ = BookmarkName$(Count)
Returns the name of the specified bookmark.

Count	The number of the bookmark as specified by its order in the list of all bookmarks, in the range 1 to CountBookmarks().

Example

```
                                    'Define array for bookmark names.
NumBookMarks = CountBookmarks()
Dim mark$(NumBookMarks)
For n = 1 To NumBookMarks           'For each bookmark,
  mark$(n) = BookmarkName$(n)       'Put bookmark name into array.
  Insert mark$(n)                   'Insert the name.
  InsertPara                        'Each on a new line.
Next
```

Transfers a list of the names of all bookmarks to the array mark$, and enters the list at the insertion point. The order of bookmark names is determined by the order of the bookmarks in the document.

_____
See also
Bookmark Statements and Functions

Call

[Call] SubName [ParameterList]
Transfers control to a subroutine. The use of Call is optional; new users of WordBasic may use it to prevent confusing a WordBasic keyword with a subroutine name while reading programs.

Example
Word executes the following two instructions identically:

```
Call FindName                    'Call the subroutine FindName.
FindName                         'Call the subroutine FindName.
```

Example

```
Sub MAIN
  If IsDirty() Then
    Call BeepThreeTimes          'Call subroutine BeepThreeTimes.
    AlertMsg                     'Call subroutine AlertMsg.
  End If
End Sub

Sub BeepThreeTimes               'Beginning of subroutine.
  For count = 1 To 3
    Beep
    For timer = 1 To 100         'Delay loop between beeps.
    Next timer
  Next count
End Sub                          'End of subroutine.

Sub AlertMsg                     'Beginning of subroutine.
  MsgBox "Document Has Changed Since Last Save", 48
End Sub                          'End of subroutine.
```

_____
See also
Standard Basic Statements and Functions

Cancel

Cancel
Terminates a mode (for example, ColumnSelect, CopyText, and CopyFormat) and does not perform the action.

_____
See also
Environment Statements and Functions
OK

CancelButton

CancelButton x, y, dx, dy
Used as part of a user dialog definition, creates a Cancel button that the user chooses to terminate the dialog box.

x, y	The coordinates of the upper-left corner of the Cancel button in increments of 1/8th (for x) and 1/12th (for y) of the system font, relative to the upper-left corner of the dialog box.
dx, dy	The width and height of the Cancel button, in increments of 1/8th (for dx) and 1/12th (for dy) of the system font.
	If the user chooses the Cancel button, an error is generated. This error can be trapped with On Error. If the function form of the dialog box is called, the function returns -1 rather than generates an error.

For more information, see "Using Macros" in the Microsoft Word User's Guide.

_____
See also
Dialog Box Statements and Functions
Dialog
Err
Error
OKButton
On Error
PushButton

CenterPara

CenterPara

n = CenterPara()
Centers the selected paragraph(s).

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of alignments.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

ChangeCase

ChangeCase [Type]
n = ChangeCase()
Sets the case of the selected text to lowercase, uppercase, or initial capitals.

Type	Specifies the change in case:
omitted	Alternates the case of the selection among all lowercase, all uppercase, and initial capitals based on the first two characters of the selection.
0 (zero)	Sets the text to all lowercase.
1	Sets the text to all uppercase.
2	Sets the text to initial capitals.

If the selection is an insertion point, Word selects the word nearest the insertion point and then changes the case of the selected text.

The function form returns:

0 (zero)	If none of the selected text is in uppercase.
1	If all of the selected text is in uppercase.
2	If the text is in a mixture of uppercase and lowercase.

Note that ChangeCase does not change the character formats associated with the selected text, as do the Small Caps or All Caps character formats.

_____

See also
Editing Statements and Functions
Formatting Statements and Functions
AllCaps
LCase$()
SmallCaps
UCase$()

ChangeRulerMode

ChangeRulerMode
Cycles the ruler between paragraph scale and margin scale. If the insertion point or selection is in a table, cycles among paragraph scale, margin scale, and table scale.

_____

See also
View Statements and Functions
RulerMode

CharColor

CharColor Color
n = CharColor()
Sets the character color of the selection to the specified color.

Color	A numeric code for the color:
0	Auto (color specified by the Control Panel setting)
1	Black
2	Blue
3	Cyan
4	Green
5	Magenta
6	Red
7	Yellow
8	White
9	Dk Blue
10	Cyan
11	Green
12	Magenta
13	Red
14	Yellow
15	Gray
16	Lt Gray

The function form returns the same number codes set by CharColor, or -1 if all the selected text is not the same color.

_____

See also

CharLeft

CharLeft [Count,] [Select]
logical = CharLeft([Count,] [Select])
Moves the insertion point left by the specified number of characters.

Count	The number of characters to move; if omitted or 0, 1 is assumed. Negative values are converted to positive values.
Select	If nonzero, the selection is extended. If there is a selection, CharLeft 1 changes the selection to an insertion point positioned at the left edge of the original selection.

The function form returns:

0	If the action cannot be performed.
-1	If the action can be performed.

Example

CharLeft 5, 1
Extends the selection five characters to the left of the insertion point.

_____
See also
Selection Statements and Functions

CharRight

CharRight [Count,] [Select]
logical = CharRight([Count,] [Select])
Moves the insertion point right by the specified number of characters.

Count	The number of characters to move; if omitted or 0, 1 is assumed. Negative values are converted to positive values.
Select	If nonzero, the selection is extended. If 0 (zero) or omitted, the selection is not extended. If there is a selection, CharRight 1 changes the selection to an insertion point positioned at the right edge of the original selection.

The function form returns:

0 (zero)	If the action cannot be performed.
-1	If the action can be performed.

Example

CharRight 10
If there is no selection, moves the insertion point 10 characters to the right of the original insertion point. If there is a selection, moves the insertion point 9 characters to the right.

_____
See also
Selection Statements and Functions

ChDir

ChDir Name$
Sets the current directory to the drive and directory specified by Name$. If the drive is omitted, the search for the specified path starts at the current directory.

Example
ChDir "c:\windows\excel\julyrpt"

_____
See also
File Statements and Functions

CheckBox

CheckBox x, y, dx, dy, Text$, .Field
Creates a check box within a dialog box.

x, y	The coordinates of the upper-left corner of the rectangle containing the check box and its associated label, in increments of 1/8th (for x) and 1/12th (for y) of the system font, relative to the upper-left corner of the dialog box.
dx, dy	The width and height of the check box, in increments of 1/8th (for dx) and 1/12th for dy) of the system font.
Text$	The associated text label for the check box. An ampersand (&) preceding a character in Text$ makes that character the underlined access key for selecting and clearing the check box.
.Field	When the dialog box that contains the check box is used, specifies a variable that returns the state of the check box:

0	The check box is not checked.
1	The check box is checked.
-1	The check box is grayed.

Example
CheckBox 97, 54, 36, 12, "&Bold", .bold
Adds a check box to a dialog box definition.
For more information, see "Using Macros" in the Microsoft Word User's Guide.

_____
See also
Dialog Box Statements and Functions
Begin Dialog...End Dialog

Chr$()

a$ = Chr$(CharCode)
Returns the character whose ANSI character code is CharCode. The following table lists a few of the special characters that can be returned using Chr$( ).

Value	Character inserted
Chr$(9)	Tab
Chr$(11)	Newline character (SHIFT+ENTER)
Chr$(13)+Chr$(10)	Paragraph mark. You should use this character sequence only to compare text in a document; to insert a paragraph mark, use InsertPara instead.
Chr$(30)	Nonbreaking hyphen
Chr$(31)	Optional hyphen
Chr$(34)	Quotation mark
Chr$(160)	Nonbreaking space

The appearance of the symbol assigned to a given character code varies with the font used, particularly for character codes greater than 127.

Examples
Chr$(67)
Returns the one-character string "C" from the numeric ANSI code specified.
Print "Title:" + Chr$(9) + Chr$(34) + "Executive Director" + Chr$(34)
Results in the text string Title:"Executive Director".

_____
See also
Standard Basic Statements and Functions

Close

Close [#StreamNumber]
Closes an open serial file specified by the file attached to StreamNumber. If StreamNumber is omitted, all open files are closed.

Examples
Close
Closes all open files.
Close #2
Closes file attached to stream number 2.
Close # handle
Closes the file attached to the stream indicated by the value of handle.

_____
See also
File I/O Statements and Functions
Open...For...As

ClosePane

ClosePane
Closes the lower window pane. Use this statement to close a pane in a split document, a header/footer pane, a footnote pane, and so on. This statement does not close a document window, only a pane in a window.

_____
See also
Window Statements and Functions

CloseUpPara

CloseUpPara
Sets to 0 (zero) the Spacing Before option (in the Format Paragraph dialog box) for the selected paragraphs.

_____
See also
Formatting Statements and Functions

CmpBookmarks()

n = CmpBookmarks(Bookmark1$, Bookmark2$)
Compares the text contained in two bookmarks.

Bookmark1$	The first bookmark.
Bookmark2$	The second bookmark.
	The function returns:
0	Bookmark1$ and Bookmark2$ are equivalent.
1	Bookmark1$ is entirely below Bookmark2$.
2	Bookmark1$ is entirely above Bookmark2$.
3	Bookmark1$ is below and inside Bookmark2$.
4	Bookmark1$ is inside and above Bookmark2$.
5	Bookmark1$ encloses Bookmark2$.
6	Bookmark2$ encloses Bookmark1$.
7	Bookmark1$ and Bookmark2$ begin at the same point, but Bookmark1$ is longer.
8	Bookmark1$ and Bookmark2$ begin at the same point, but Bookmark2$ is longer.
9	Bookmark1$ and Bookmark2$ end at the same place, but Bookmark1$ is longer.
10	Bookmark1$ and Bookmark2$ end at the same place, but Bookmark2$ is longer.
11	Bookmark1$ is below and adjacent to Bookmark2$.
12	Bookmark1$ is above and adjacent to Bookmark2$.
13	One or more of the bookmarks do not exist.

_____
See also
Bookmark Statements and Functions

ColumnSelect

ColumnSelect
Starts the column selection mode. Cancel, OK, and any command acting on the column selection ends this mode.

_____
See also
Selection Statements and Functions

ComboBox

ComboBox x, y, dx, dy, ArrayVariable$(), .Field
Creates an expanded combo box.

x, y	The coordinates of the upper-left corner of the Cancel button, in increments of 1/8th and 1/12th the system font, relative to the upper-left corner of the dialog box.
dx, dy	The width and height of the Cancel button, in increments of 1/8th and 1/12th the system font.
ArrayVariable$()	A text array containing the items to be listed in the combo box.
.Field	A variable returning the text of the item chosen.
	For more information, see "Using Macros" in the Microsoft Word User's Guide.

Example
ComboBox 6, 31, 49, 33, Arr$(), .LB

_____
See also
Dialog Box Statements and Functions

CommandValid()

n = CommandValid(CommandName$)
Returns -1 if CommandName$ is both a dialog box equivalent and valid for the context in which it is being used.

_____
See also

ControlRun

ControlRun .Application = number
Corresponds to the Run dialog box (application Control menu); runs an application listed in the Run dialog box.

Application	The number of the application as listed in the Run dialog box.

Examples
ControlRun .Application = 0
Runs the Clipboard.
ControlRun .Application = 1
Runs the Control Panel.

_____
See also
Environment Statements and Functions

CopyBookmark

CopyBookmark Bookmark1$, Bookmark2$
Sets Bookmark2$ equal to Bookmark1$.

Example
CopyBookmark "sales_receipts", "month_sales"
Causes the bookmark month_sales to be associated with the same text as the bookmark sales_receipts.

_____
See also
Bookmark Statements and Functions

CopyFile

CopyFile Filename$, Directory$
Copies a file to the specified directory. If a file of the same name already exists in the specified directory, a message appears asking if you want to replace the existing file.

Filename$	The name of the file to copy. If you do not specify a drive and directory, Word copies the file from the current directory.
Directory$	The full path of the directory to which the file is copied.

_____
See also
File Statements and Functions

CopyFormat

CopyFormat
Copies the formatting of the selected text to another range of selected text. To use CopyFormat, include or record instructions corresponding to the following steps:

1	Make a selection.
2	Run CopyFormat.
3	Select where you want to paste the formats.
4	Run OK (press ENTER).

Example

```
StartOfLine                    'Move insertion point to start of current line
EndOfLine 1                    'Move insertion point to end of line while selecting
CopyFormat                     'Copy formatting

LineUp                         'Move ins. pt. to previous line.
StartOfLine                    'Move ins. pt. to start of line.
EndOfLine 1                    'Move ins. pt. to end of line while selecting.
OK                             'Apply copied format to selection.
```

_____
See also
Formatting Statements and Functions

CopyText

CopyText
Copies selected text without putting it on the Clipboard; equivalent to pressing SHIFT+F2. To use CopyText, include or record instructions corresponding to the following steps:

1	Make a selection.
2	Run CopyText.
3	Select where you want to paste the text.
4	Run OK (press ENTER).

_____

See also
Editing Statements and Functions

CountBookmarks()

n = CountBookmarks()
Returns the number of bookmarks defined in the active document.

_____
See also
Bookmark Statements and Functions

CountFiles()

n = CountFiles()
Returns the number of names in the file list at the bottom of the File menu.

Example

```
For count = 1 To CountFiles()
  FileArray$(count) = FileName$(count)
Next
```

Loads the names of all files at the bottom of the File menu into a string array called FileArray$.

_____
See also
File Statements and Functions

CountFonts()

n = CountFonts()
Returns the number of fonts available with the selected printer. This is the number of the fonts that appear both in the Character dialog box (Format menu) and on the ribbon's font list.

Example
```
Dim FontName$(CountFonts())
For count = 1 To CountFonts()
  FontName$(count) = Font$(count)
Next
```
Loads the names of all current printer fonts into a string array called FontName$.

_____
See also
Formatting Statements and Functions

CountFoundFiles()

n = CountFoundFiles()
Returns the number of files found in the last Find File search.

_____
See also
File Statements and Functions

CountGlossaries()

n = CountGlossaries([Context])
Returns the number of glossaries defined for the specified context.

Context	Scope of application:
0 (zero)	Global (default)
1	Document template

_____
See also
Glossary Statements and Functions

CountKeys()

n = CountKeys([Context])
Returns the number of key assignments in the Keyboard category of the Options dialog box (Tools menu) that differ from the default assigments.

Context	Scope of application:
0 (zero)	Global (default)
1	Document template

Example
```
tab$ = Chr$(9)
For n = 1 to CountKeys()
  Insert Str$(n) + tab$ + KeyMacro$(n) + tab$ + Str$(KeyCode(n))
  InsertPara
Next
```
Inserts a tab-delimited list of the currently defined shortcut key sequences into the active document.

_____
See also
Macro Statements and Functions

CountLanguages()

n = CountLanguages()
Returns the number of languages listed in the Language dialog box (Format menu), including the No Proofing option.

_____
See also
Macro Statements and Functions

CountMacros()

n = CountMacros([Context,] [All])
Returns the number of macros defined for the specified context.

Context	**Scope of application:**
0 or omitted	Global
1	Document template
All	If nonzero, built-in commands are included in the count.

Example

```
NumBuiltIns = CountMacros(0,1) - CountMacros(0)
For n = 1 to NumBuiltIns
  Insert Str$(n) + Chr$(9) + MacroName$(n,0,1)
  InsertPara
Next
```

This routine stores the number of built-in commands in NumBuiltIns and then inserts a list of the names of the built-in commands in the active document.

_____
See also
Macro Statements and Functions

CountMenuItems()

n = CountMenuItems(MenuNumber, Context)
Returns the number of menu items on the specified menu that differ from the default assignments.

MenuNumber The number of the menu, with the File menu being number 1, Edit 2, and so on.

Context	Scope of application:
0 or omitted	Global
1	Document template

See also
Macro Statements and Functions

CountMergeFields()

n = CountMergeFields()
Returns the number of fields in the header record of the data file or header file associated with the active merge document.

See also
File Statements and Functions
MergeFieldName$()

CountStyles()

n = CountStyles([Context,] [All])
Returns the number of styles defined for the specified context.

Context	Scope of application:
0 or omitted	Active document; includes styles defined in the active document and the active template.
1	Document template; includes styles defined in the active template.
All	If nonzero, unused standard styles are included. Word contains 34 standard styles. If 0 (zero) or omitted, standard styles are excluded.

Examples

n = CountStyles(1,1)
Returns the number of all built-in styles plus the number of user-created styles used in the template attached to the active document.
n = CountStyles(0) - CountStyles(1)
Returns the number of styles defined specifically for the active document.

See also
Formatting Statements and Functions

CountWindows()

n = CountWindows()
Returns the number of windows listed on the Window menu.

See also
Window Statements and Functions

Date$()

a$ = Date$()
Returns today's date. The date format is determined by the date setting in the operating environment's control panel.

See also
Standard Basic Statements and Functions

DDEExecute

DDEExecute ChanNum, ExecuteString$
Sends an execute message over the specified channel.

ChanNum The channel number of the application to which the message is sent, as returned by the

	DDEInitiate() function when the channel is opened.
ExecuteString$	A message that is defined in the receiving application. Use the format described under SendKeys to send specific key sequences. If the channel is not valid or if the receiving application refuses to execute the instructions, an error is generated.

_____

See also

Dynamic Data Exchange (DDE) Statements and Functions

DDEInitiate()

ChanNum = DDEInitiate(App$, Topic$)
Opens a DDE channel to an application.

App$	The application name defined by the other application.
Topic$	Describes something in the application you are accessing, usually the document that contains the data you want to use.
	If DDEInitiate() is successful, it returns the number of the open channel. All subsequent DDE functions use this number to specify the channel. This function returns 0 (zero) if it fails to open a channel.

Example
ChanNum = DDEInitiate("EXCEL", "LOAN.XLS")
Initiates a channel to Microsoft Excel and the file LOAN.XLS; ChanNum holds the channel number returned.

_____
See also
Dynamic Data Exchange (DDE) Statements and Functions

DDEPoke

DDEPoke ChanNum, Item$, Data$
Sends data to an application.

ChanNum	The channel number of the application as returned by DDEInitiate().
Item$	The item to which the data is to be sent.
Data$	The data to send to the application.
	The channel must have been opened by the DDEInitiate() function. If DDEPoke is unsuccessful, an

error is generated.

_____
See also
Dynamic Data Exchange (DDE) Statements and Functions

DDERequest$()

a$ = DDERequest$(ChanNum, Item$)
Requests the information from the specified application.

ChanNum	The channel number of the application as returned by DDEInitiate.
Item$	Specifies the location of the information requested.
	The channel must have been opened by the DDEInitiate() function.

This function returns data in the CF_TEXT format; if unsuccessful, returns a null string (""). Pictures or text in rich-text format cannot be transferred.

_____
See also
Dynamic Data Exchange (DDE) Statements and Functions

DDETerminate

DDETerminate ChanNum
Closes a channel to another application.

ChanNum	The channel number of the application as returned by DDEInitiate().
	The channel must have been opened with the DDEInitiate() function.

_____
See also
Dynamic Data Exchange (DDE) Statements and Functions

DDETerminateAll

DDETerminateAll
Closes all open channels.

_____

See also

Dynamic Data Exchange (DDE) Statements and Functions

DDETerminate

Declare

Declare Sub SubName Lib LibName [ParameterList[$] As String (or Integer, Double, or Long)] [Alias ModuleName]
Declare Function FunctionName Lib LibName [ParameterList] [Alias ModuleName] As String (or Integer, Double, or Long)
Declares an external library function as a subroutine or function inside a macro. Parameters not specified as string or integer default to floating point. Use of this feature requires knowledge of Windows programming.

Example

```
Declare Function IsAppLoaded Lib "kernel" (name$) As Integer Alias "GetModuleHandle"
Sub MAIN
  If IsAppLoaded("EXCEL") = 0 Then
    MsgBox "Excel is not running"
  Else
    MsgBox "Excel is loaded"
  End If
End Sub
```

_____
See also
Standard Basic Statements and Functions

DeleteBackWord

DeleteBackWord
Deletes the word immediately preceding the selection but does not place it on the Clipboard; the equivalent of pressing CTRL+BACKSPACE.

_____
See also
Editing Statements and Functions

DeleteWord

DeleteWord
Deletes the word immediately following the insertion point or the first word included in the selection but does not place it on the Clipboard, leaving the Clipboard undisturbed. The equivalent of pressing CTRL+DEL.

_____
See also
Editing Statements and Functions

Dialog

Dialog DialogRecord
n = Dialog(DialogRecord)
Displays for data entry the dialog box specified by DialogRecord. After editing, you can store edits in DialogRecord by choosing OK, or you can lose edits by choosing Cancel. Choosing Cancel produces a run-time error that you can trap with On Error. If the function form of the dialog box is called, the function returns -1 rather than generating an error.

The function form returns:

-1	If the OK button is chosen.
0	If the Cancel button is chosen.
>0	If a command button is chosen. The button number is determined by the position of the corresponding PushButton statement in the dialog declaration: 1 for the first PushButton, 2 for the second, and so on. For more information, see "Using Macros" in the Microsoft Word User's Guide.

_____
See also
Dialog Box Statements and Functions

Dim

Dim [Shared] Var [(Size)] [, Var [(Size)]]
Dim rec As DialogRecord
Declares an array or nonarray variable and allocates storage for an array. Using the Shared keyword permits sharing a variable and its value(s) among the routines in a macro.
Size is the number of the last element in the array. When defining array variables for items in a list box, make sure you define the first item as 0 (zero) -- for example, ListBoxItem$(0) = "Apple".
The second form is used in defining dialog records; for more information, see "Using Macros" in the Microsoft Word User's Guide.

Example

```
Dim Months$(12)
Month$(1) = "January"
Month$(2) = "February"
Month$(3) = "March"
etc.
Print Month$(2)
```

Stores the names of the months in the array Months$, and displays "February" in the status bar.

```
Dim Shared num                          'Declare num as shared.
Sub Main
  MyRoutine                             'Call the routine.
  Print num                             'Display num in status bar.
End Sub

Sub MyRoutine
  num = 123                             'Set the value of num.
End Sub
```

Declares the variable num as shared, before the first line in the Main routine, sets the variable to the value 123 in MyRoutine, and displays the number in the status bar.

_____
See also
Standard Basic Statements and Functions

DisableAutoMacros

DisableAutoMacros [Disable]
Disables subsequent execution of AutoExec, AutoOpen, AutoClose, AutoNew, and AutoExit macros until re-enabled or Word is started again.

Disable	**Specifies whether or not to disable auto macros:**
0	Enables auto macros.
1 or omitted	Disables auto macros.

_____
See also
Macro Statements and Functions

DisableInput

DisableInput [Disable]
Prevents the ESC key from interrupting a macro; does not affect the use of ESC for cancelling a dialog box.

Disable	**Determines whether or not to disable the ESC key:**
0	Enables the ESC key.
1 or omitted	Disables the ESC key.

Examples

```
DisableInput 0      'Disable inactive
DisableInput 1      'Disable active
```

_____
See also
Macro Statements and Functions

DocClose

DocClose [Save]
Closes the active document window or pane.

Save	**Determines whether or not to save the document:**
0 or omitted	Prompts the user to save if the document is "dirty" (that is, changes have been made since the

	last time the document was saved).
1	Word saves the document without prompting.
2	Word closes the window or pane but does not save the document.

_____

See also
File Statements and Functions
Window Statements and Functions

DocMaximize

DocMaximize
logical = DocMaximize()
Zooms the document window to the maximum size available for the application. If it is already maximized, the window is displayed in the restored state.

The function form returns:

-1	If the window is maximized.
0	If the window is not maximized.

Example

If Not DocMaximize() Then DocMaximize
Tests the document window to see if it is maximized. If the window is maximized, no action is performed; if the window is not maximized, maximizes the window.

_____
See also
Window Statements and Functions

DocMove

DocMove XPos, YPos
Moves the document window to the specified location. Word cannot perform this action if the document window is maximized.

XPos, YPos	The coordinates of the upper-left corner of the window, relative to the upper-left corner of the Word workspace. Values are given in points.

Example

DocMove 20, 40
Moves the window to the position 20 points to the right and 40 points down from the upper-left corner of the document area.

_____
See also
Window Statements and Functions

DocRestore

DocRestore
Restores the document window from a maximized state.

_____
See also
Window Statements and Functions

DocSize

DocSize Width, Height
Sizes the document window to the specified Width and Height, in points. Word cannot perform this action if the document window is maximized.

_____
See also
Window Statements and Functions

DocSplit

DocSplit Percentage
n = DocSplit()
Splits the active window at the given height, expressed as a percentage of the distance between the top and bottom of the active window.
The function form returns the split position as a percentage of the window height, or returns 0 (zero) if the window is not split.

Examples
DocSplit 50
Splits the active window in the middle.
If DocSplit() > 50 Then DocSplit 50
If the active window is split more than 50% of the way down its height, then the window is split in the middle.

_____
See also
Window Statements and Functions

DocumentStatistics

DocumentStatistics .FileName = text, .Directory = text, .Template = text, .Title = text, .Created = text, .LastSaved = text, .LastSavedBy = text, .Revision = number, .Time = text, .Printed = text, .Pages = number, .Words = number, .Characters = number, .Update
Corresponds to the dialog box that appears when the user clicks the Statistics button in the Summary Info dialog box (File menu); sets and returns information about the specified document. The following arguments are read-only.

.FileName	The name of the document.
.Directory	The full path of the directory in which the document is kept.
.Template	The full path of the template associated with the document.
.Title	The document's title.
.Created	The date and time the document was created.
.LastSaved	The date and time the document was last saved.
.LastSavedBy	The author of the document.
.Revision	The current number of revisions.
.Time	The total time spent editing the document.
.Printed	The date and time the document was last printed.
.Pages	The number of pages in the document.
.Words	The number of words in the document.
.Characters	The number of characters in the document.
.Update	Updates the values of Pages, Words, and Characters.

Example

```
Sub MAIN
  Dim dlg As DocumentStatistics
  GetCurValues dlg
  Print "Words in document: " + dlg.Words
End Sub
```

Displays the current number of words in the document in the status bar.

Note The main difference between this statement and FileSummaryInfo is that DocumentStatistics updates its information before returning it to the macro. With FileSummaryInfo, your macro must call the statement twice -- once using the .Update argument to update the information, and once to get the information.

_____

See also
File Statements and Functions
FileSummaryInfo
SelInfo

DoFieldClick

DoFieldClick
Simulates the double-clicking of a mouse button or pressing ALT+SHIFT+F9 within a GOTOBUTTON or MACROBUTTON field.

_____

See also
Field Statements and Functions

DoubleUnderline

DoubleUnderline [On]
n = DoubleUnderline()
Adds or removes the double underline character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

EditClear

EditClear [Count]
Deletes the selection without changing the contents of the Clipboard. If the selection is an insertion point, deletes the character to the right of the insertion point.

Count	**The number of characters to delete:**
>0	Deletes the specified number of characters to the right of the insertion point.
0 or omitted	Deletes the selection or the character to the right of the insertion point.
<0	Deletes the specified number of characters to the left of the insertion point.

Note: If Count is nonzero and there is a selection, Word treats the selection as the first character to delete.

_____

See also
Editing Statements and Functions

EditCopy

EditCopy
Corresponds to the Copy command on the Edit menu; copies the selection to the Clipboard.

_____
See also
Editing Statements and Functions

EditCut

EditCut
Corresponds to the Cut command on the Edit menu; places the selection on the Clipboard and then deletes it from the document.

_____
See also
Editing Statements and Functions

EditFind

EditFind [.Find = text,] [.WholeWord = number,] [.MatchCase = number,] [.Direction = number,] [.Format = number]
Corresponds to the Find dialog box (Edit menu); finds the specified text, given a search string. If an argument is not specified, the search is performed with the last-specified options.

.Find	The text to search for, or a search string.
.WholeWord	Corresponds to the Whole Word check box.
.MatchCase	Corresponds to the Match Upper/Lowercase check box.
.Direction	Direction to search:
0 (zero)	Searches toward beginning of document, without prompt.
1	Searches toward end of document, with prompt.
2	Searches toward end of document, without prompt.
	The default is the last direction argument used, or 1 the first time the Find command is executed.
.Format	Finds formatting in addition to, or instead of, text:
0 (zero)	Ignore formatting (default).
1	Use the formatting specified by EditFindChar, EditFindPara, and/or EditFindStyle.

Examples
EditFind .Find="Trey Research" .Direction=1
Searches toward the end of the document for the text "Trey Research".
EditFindChar .Underline = 1
EditFind .Format = 1
Finds the first instance in the specified direction of underlined text in the document.

_____
See also
Editing Statements and Functions

EditFindChar

EditFindChar [.Font = text,] [.Points = value,] [.Bold = number,] [.Italic = number,] [.Strikeout = number,] [.Hidden = number,] [.SmallCaps = number,] [.AllCaps = number,] [.Underline = number,] [.Color = number,] [.Position = value,] [.Spacing = value,] [.UseAsDefault]
Corresponds to choosing the Character button in the Find or Replace dialog box (Edit menu); defines the character formatting used to find formatted text. The arguments specify options available in the Character dialog box (Format menu).

Examples
EditFindChar .Bold = 0
Finds instances of specific text that are not bold.
EditFindChar .Bold = 1
Finds instances of specific text that are bold.
EditFindChar .Bold = -1
Finds instances of specific text that are either bold or not bold.

_____
See also
Editing Statements and Functions
Formatting Statements and Functions
EditFind
EditReplace

EditFindClearFormatting

EditFindClearFormatting
Corresponds to choosing the Clear button in the Find or Replace dialog box (Edit menu); clears the formats used to find formatted text.

_____
See also
Editing Statements and Functions
Formatting Statements and Functions

EditFindFound()

logical = EditFindFound()
Returns a value indicating whether or not the last EditFind was successful:

-1	The last EditFind was successful.
0 (zero)	The last EditFind was not successful.

Example

```
Sub Main
  count = 0
  StartOfDocument
  EditFind .Find = "macro", .Direction = 2
  While EditFindFound()
    count = count + 1
    EditFind
  Wend
  Print "macro was found "; Str$(count); " times"
End Sub
```

_____
See also
Editing Statements and Functions

EditFindPara

EditFindPara [.Alignment = number,] [.LeftIndent = text,] [.RightIndent = text,] [.FirstIndent = text,] [.Before = text,] [.After = text,] [.LineSpacing = text,] [.PageBreak = number,] [.KeepWithNext = number,] [.KeepTogether = number,] [.NoLineNum = number]
Corresponds to choosing the Paragraph button in the Find or Replace dialog box (Edit menu); defines the paragraph formatting used to find formatted text. The arguments specify options available in the Paragraph dialog box (Format menu).

Example
EditFindPara .Alignment = 0
Finds the next occurrence of a paragraph having left alignment.

_____
See also
Editing Statements and Functions
Formatting Statements and Functions
EditFind

EditFindStyle

EditFindStyle .Style = text
Corresponds to setting options in the Style dialog box, brought up by choosing the Styles button in the Find or Replace dialog box (Edit menu); finds text having the specified style.

.Style	The name of a style; to specify "no style", use an empty string.

Example
EditFindStyle .Style = "Normal"
EditFind .Find = "",.Format = 1
Finds and selects the next paragraph having the Normal style.

_____
See also
Editing Statements and Functions
Formatting Statements and Functions

EditFootnoteContNotice

EditFootnoteContNotice
Corresponds to clicking the Cont. Notice button in the Footnote Options dialog box; opens a pane for editing the continuation notice for footnotes.

_____
See also
Editing Statements and Functions

EditFootnoteContSep

EditFootnoteContSep
Corresponds to clicking the Cont. Separator button in the Footnote Options dialog box; opens a pane for editing the continuation separator for footnotes.

_____
See also
Editing Statements and Functions

EditFootnoteSep

EditFootnoteSep
Corresponds to clicking the Separator button in the Footnote Options dialog box; opens a pane for editing the separator for footnotes.

_____
See also
Editing Statements and Functions

EditGlossary

EditGlossary .Name = text, [.Context = number,] .Insert, .InsertAsText, .Define, .Delete
Corresponds to the Glossary dialog box (Edit menu); used to define, delete, and insert glossary entries.

.Name	A name of a glossary entry.
.Context	The context of the glossary entry, used only when .Define is used:
0 or omitted	Global
1	Template
.Insert	The default action.
.InsertAsText	Corresponds to clicking the Insert As Plain Text button.
	Only one of the following arguments can be used at a time:
.Insert	The default action.
.InsertAsText	Corresponds to clicking the Insert As Plain Text button.
.Define	Defines the entry.
.Delete	Deletes the entry.

Example

EditGlossary .Name = "CPR"
Inserts the glossary entry named "CPR" at the insertion point.
EditGlossary .Name = "CPR", .Define
Defines the global glossary entry named "CPR".
EditGlossary .Name = "CPR", .Delete
Deletes the global glossary entry named "CPR".

_____
See also
Editing Statements and Functions

EditGoTo

EditGoTo .Destination = text
Corresponds to the Go To dialog box (Edit menu) and the F5 key; jumps to the specified bookmark or goto string.

.Destination	A bookmark name or goto string.

Examples

EditGoTo .Destination = "5"
Goes to page 5.
EditGoTo .Destination = "s1"
Goes to section 1.
EditGoTo .Destination = "l+10"
Moves the insertion point 10 lines down.
EditGoTo .Destination = "s1l50"
Moves the insertion point to line 50 of section 1.
EditGoTo .Destination = "%50"
Moves the insertion point to the middle of the document.
EditGoTo .Destination = "Disclaim"
Moves the insertion point to the bookmark "Disclaim".
EditGoTo .Destination = "f1"
Moves the insertion point to footnote 1.
EditGoTo .Destination = "\StartOfDoc"
Uses a special bookmark to move the insertion point to the beginning of the document.
EditGoTo .Destination = "a"
Moves the insertion point to the next annotation.

_____
See also
Bookmark Statements and Functions
Selection Statements and Functions

EditLinks

EditLinks .UpdateMode = number, .Locked = number, .OpenSource, .UpdateNow, .KillLink, .Link = text, .Application = text, .Item = text, .Filename = text
Corresponds to the Links dialog box (Edit menu); sets parameters for the specified link.

.UpdateMode	Corresponds to the Update option:
1	Automatic
2	Manual

.Locked	Corresponds to the Locked check box:
0	Unlocks the link.
1	Locks the link.

.OpenSource	Corresponds to clicking the Open Source button.
.UpdateNow	Corresponds to clicking the Update Now button.
.KillLink	Corresponds to clicking the Cancel Link button.
.Link	The name of the link.
.Application	The name of the application supplying the link.
.Item	The item in the application that supplies the link; usually the name of the document.
.FileName	The full path of the document supplying the link in the originating application.
	The last three arguments correspond to clicking the Change Link button and setting options in the Change Link dialog box.

Example
EditLink .UpdateNow, .Link = "1", .Application = "WinWord",\
.Item = "INTERN_LINK2", .Filename = "C:\MYDOCS\TESTLINK.DOC"
Updates a link to the open document TESTLINK.DOC.

_____
See also
Dynamic Data Exchange (DDE) Statements and Functions

EditObject

EditObject
Opens the selected object for editing in the object's native application.

_____
See also
Editing Statements and Functions

EditPaste

EditPaste
Corresponds to the Paste command on the Edit menu; inserts the contents of the Clipboard at the insertion point. Replaces the selection if you select Typing Replaces Selection, a General option in the Options dialog box (Tools menu).

_____
See also
Editing Statements and Functions

EditPasteSpecial

EditPasteSpecial [.Link = number,] [.DataType = text]
Corresponds to the Paste Special command on the Edit menu; pastes information from the Clipboard at the insertion point or selection.

.Link	A number specifying the data type of the link to paste:
0 or omitted	Paste the data using the data format specified in .DataType.
1	Create a link using the data format specified in .DataType.

.DataType	Specifies the type of data on the Clipboard:
Text	Unformatted text
RTF	Rich-text format
Pict	Windows metafile
Bitmap	Windows bitmap
Object	OLE object format
DIB	Windows Device Independent Bitmap

_____
See also
Editing Statements and Functions

EditPic

EditPic
Brings up the Draw application for editing the selected graphic.

_____
See also
Editing Statements and Functions

EditRepeat

EditRepeat
Corresponds to the Repeat command on the Edit menu; repeats the last editing operation, if possible.

_____
See also
Editing Statements and Functions

EditReplace

EditReplace [.Find = text,] [.Replace = text,] [.WholeWord = number,] [.MatchCase = number,] [.Format = number,] [.ReplaceAll]
Corresponds to the Replace dialog box (Edit menu); replaces one text string with another.

.Find	The text to find.
.Replace	The text to replace. If Find and Replace are omitted, the strings used in the previous search and/or replace are used.
.WholeWord	Corresponds to the Whole Word check box.
.MatchCase	Corresponds to the Match Case check box.

.Format	**Replaces formatting in addition to, or instead of, text:**
0	Does not replace formatting.
1	Does replace formatting.

.ReplaceAll	Corresponds to clicking the Replace All button.
	To replace formatting in addition to, or instead of, text, first use the EditFindChar, EditFindPara, EditFindStyle, EditReplaceChar, EditReplacePara, EditReplaceStyle, EditFindClearFormatting, or EditReplaceClearFormatting instructions to set up the formatting, and then use EditReplace or EditFind with Format set to 1.

Example
EditFindChar .Underline = 1
EditReplaceChar .Italic = 1
EditReplace .ReplaceAll
EditReplaceClearFormatting
Finds all instances of underlined text in the document and replaces the format with italics.

_____
See also
Editing Statements and Functions

EditReplaceChar

EditReplaceChar [.Font = text,] [.Points = value,] [.Bold = number,] [.Italic = number,] [.Strikeout = number,] [.Hidden = number,] [.SmallCaps = number,] [.AllCaps = number,] [.Underline = number,] [.Color = number,] [.Position = value,] [.Spacing = value]
Corresponds to clicking the Character button in the Replace dialog box and specifying character formats in the Find Character dialog box; defines the character formatting EditReplace uses to format replacement text. The arguments specify options available in the Character dialog box (Format menu).
For a description of the arguments, see FormatCharacter.

_____
See also
Editing Statements and Functions
Formatting Statements and Functions
EditReplace

EditReplaceClearFormatting

EditReplaceClearFormatting
Corresponds to clicking the Clear button in the Replace dialog box (Edit menu); clears the formats EditReplace uses to format replacement text.

_____
See also
Editing Statements and Functions
Formatting Statements and Functions
EditReplace

EditReplacePara

EditReplacePara [.Alignment = number,] [.LeftIndent = text,] [.RightIndent = text,] [.FirstIndent = text,] [.Before = text,] [.After = text,] [.LineSpacing = text,] [.PageBreak = number,] [.KeepWithNext = number,] [.KeepTogether = number,] [.NoLineNum = number]
Corresponds to clicking the Paragraph button in the Replace dialog box (Edit menu); defines the paragraph formatting EditReplace uses to format replacement text. The arguments specify options available in the Paragraph dialog box (Format menu).

_____
See also
Editing Statements and Functions
Formatting Statements and Functions

EditFind
EditFindPara
EditReplace

EditReplaceStyle

EditReplaceStyle .Style = text
Corresponds to clicking the Styles button in the Replace dialog box (Edit menu); defines the style EditReplace uses to format replacement text.

.Style	The name of the style to replace; to specify "no style", use an empty string.

See also
Editing Statements and Functions
Formatting Statements and Functions
EditFindStyle
EditReplace

EditSelectAll

EditSelectAll
Selects the entire document.

_____
See also
Selection Statements and Functions

EditUndo

EditUndo
Corresponds to the Undo command on the Edit menu; undoes the last action, if possible. You can undo certain Word actions (for example, Cut and Paste); some actions cannot be undone. See Reversing an action.

_____
See also
Editing Statements and Functions

EmptyBookmark()

logical = EmptyBookmark(Name$)
Determines whether or not the specified bookmark is empty.

Name$	The name of the bookmark to check.

The function returns:

-1	If the bookmark is empty (an insertion point).
0	If the bookmark is not empty.

_____
See also
Bookmark Statements and Functions

EndOfColumn

EndOfColumn [Select]
logical = EndOfColumn([Select])
Moves the insertion point to the bottom cell in a table column. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the end of a column.
-1	If the action was performed.

_____
See also
Selection Statements and Functions

EndOfDocument

EndOfDocument [Select]
logical = EndOfDocument([Select])
Moves the insertion point to the end of the document. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the end of the document.
-1	If the action was performed.

_____
See also
Selection Statements and Functions

EndOfLine

EndOfLine [Select]
logical = EndOfLine ([Select])
Moves the insertion point to the end of the line. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the end of a line.
-1	If the action was performed.

_____

See also
Selection Statements and Functions

EndOfRow

EndOfRow [Select]
logical = EndOfRow([Select])
Moves the selection to the end of the last cell in the table row. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the end of a row.
-1	If the action was performed.

_____

See also
Selection Statements and Functions

EndOfWindow

EndOfWindow [Select]
logical = EndOfWindow([Select])
Moves the selection to the end of the window. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the end of a window.
-1	If the action was performed.

_____

See also
Selection Statements and Functions

Eof()

logical = Eof(StreamNumber)
Determines whether or not the end of an open serial text file has been reached.

StreamNumber	The number specified in the Open instruction that opened the text file.

The function returns:

-1	If the end of the file attached to StreamNumber has been reached.
0	If the end of the file has not been reached.

_____

See also
File I/O Statements and Functions

Err

n = Err
Err = 0
This is a special variable that contains the error code for the most recent error condition.
The form Err = 0 is used to reset error trapping after an error has occurred. This form is normally used at the end of an error trapping routine. If Err = 0 is not included, only the first error that occurs is trapped.
Future errors generate an error message and cause the macro to halt. An On Error Goto instruction also resets Err to 0.

Examples

```
Sub MAIN
  On Error Goto Trouble
  port = 1
Start:
  Insert WindowName$(port)
  InsertPara
  port = port + 1
  Goto Start
Trouble:
  Error Err
End Sub
```

This example first sets up an error trap using the On Error Goto instruction. The value of the variable port is set to 1. The name of window with the number of port (1) is printed. The value of port is increased by one and the process is repeated until the value of port exceeds the number of open windows. At this point, an error occurs and the error trap directs the program to the label Trouble. The Error instruction prints an error message for the designated error number and causes the macro to halt. The number of the last occurring error is returned from the reserved variable Err.
As shown here, with a slightly altered error routine, the preceding program could deal with the error and allow the program to continue:

```
Trouble:
  If Err = 5 Then                    'Error 5: Illegal Function Call
    port = 1                         'If found, set value of Port to 1
    Err = 0                    'Reset error trap
    Goto Done                        'Return to main program
  Else                               'If a different error occurs
    Error Err                        'Print error message and halt
Done:
End Sub
```

Errors with numbers 1,000 or greater are generated by Word and not by the programming language itself. If such an error occurs, an error message box is displayed, and the user must respond before the macro can continue. If OK is chosen, control of the program then passes to the error-handling routine.

_____

See also
Standard Basic Statements and Functions

Error

Error ErrorNumber
Generates the specified error condition.

ErrorNumber	A number specifying the error code to generate.
	Error serves two functions. It can be used to generate a specified error condition in order to test an error-handling routine, or it can be used to display an error message and terminate operation of the macro program.

Errors with numbers 1,000 or greater are generated by Word and not by the programming language itself. If such an error occurs, an error message box is displayed, and the user must respond before the macro can continue. If OK is chosen, control of the program then passes to the current On Error instruction. If an argument of 1,000 or greater is used with Error, no message box is displayed, but the
error trap (if any) is triggered.

Example
In the following example, Error is used for both purposes: first, to generate a simulated error that triggers the error-handling routine, and second, to display a message and end the macro if the error is not specifically handled.

```
Sub MAIN
  On Error Goto ErrorHandler
  'Generate error 530: "Dialog box too complex"
  Error 530

ErrorHandler:
  Select Case Err                    'Branch based on error number
  Case 5
  ...process error                   'Statements handling error 5
  Case 11
  ...process error                   'Statements handling error 11
  Case 71
  ...process error                   'Statements handling error 71
  Case Else                          'Error (i.e. 530) not handled above
    Error Err                        'Generate error message and end.
  End Select
End Sub
```

_____
See also
Standard Basic Statements and Functions

ExistingBookmark()

logical = ExistingBookmark(Bookmark$)
Used to test for the presence of a particular bookmark.

Bookmark$	The name of a bookmark.
	The function returns:
-1	If the bookmark exists.
0 (zero)	If the bookmark does not exist.

_____
See also
Bookmark Statements and Functions

ExpandGlossary

ExpandGlossary
Expands the word closest to the insertion point into the corresponding glossary text.

_____
See also
Glossary Statements and Functions

ExtendMode()

logical = ExtendMode()
Returns a nonzero value if extend mode is in effect. ExtendMode does not report whether or not column extend mode is in effect.

_____
See also
Selection Statements and Functions

ExtendSelection

ExtendSelection [Character$]
Turns on extend mode, if it is not already turned on. If extend mode is already turned on, the selection is extended to the next unit; for example, if extend mode is already on and a character is selected, ExtendSelection extends the selection to the whole word.

Character$ The character to which to extend the selection.

Use the Cancel statement to exit this mode.

_____
See also
Selection Statements and Functions

File1

File1
Corresponds to selecting the first file listed at the bottom of the File menu; that is, opens the first file. An error is generated if you attempt to open a nonexistent file slot; that is, you cannot use File1 if the File menu lists no files.

_____
See also
File Statements and Functions
CountFiles

File2

File2
Corresponds to selecting the second file listed on the File menu; that is, opens the second file. An error is generated if you attempt to open a nonexistent file slot; that is, you cannot use File2 if the File menu lists only one file or no file.

_____
See also
File Statements and Functions
CountFiles

File3

File3
Corresponds to selecting the third file listed on the File menu; that is, opens the third file. An error is generated if you attempt to open a nonexistent file slot; that is, you cannot use File3 if the File menu lists two or fewer files.

_____
See also
File Statements and Functions
CountFiles

File4

File4
Corresponds to selecting the fourth file listed on the File menu; that is, opens the fourth file. An error is generated if you attempt to open a nonexistent file slot; that is, you cannot use File4 if the File menu lists three or fewer files.

_____
See also
File Statements and Functions
CountFiles

FileClose

FileClose [Save]
Corresponds to the Close command on the File menu. Closes the active file and associated windows.
(Use DocClose to close only the active window of a document.)

Save	**Determines whether or not a save is forced:**
0 or omitted	The user is prompted to save if the file is "dirty" (that is, changes have been made since the last time the file was saved).
1	Saves the document before closing it.
2	Closes the document without saving it.

_____
See also
File Statements and Functions

FileCreateDataFile

FileCreateDataFile .FileName = text [,.PasswordDoc = text] [,.PasswordDot = text] [,.FieldName = text]
Corresponds to clicking the Create Data File button in the Select Data File dialog box; creates a new data document for a print merge document, or adds a field to a data file.

.FileName	The full path of the new data file.
.PasswordDoc	The password for the file.
.PasswordDot	The password for the document's DOT file.
.FieldName	The name of a field or fields to add to the data file, as text.

Example

```
FileCreateDataFile .FileName = "C:\WORDDATA\MYDATA.DOC", \
   .FieldName = "name, address, city, state, postcode"
```

Creates a new data document associated with the active document, which contains the merge fields name, address, city, state, and postcode, and opens a new window containing a table listing these merge fields.

_____

See also

Merge Statements and Functions

FileCreateHeaderFile

FileNewHeaderFile .FileName = text [,.PasswordDoc = text] [,.PasswordDot = text] [,.FieldName = text]
Creates a new header document for a print merge document.

.FileName	A filename for the new document.
.PasswordDoc	The password for the header document.
.PasswordDot	The password for the document's DOT file.
.FieldName	The name of a field or fields to add to the data file, as text.

_____

See also
Merge Statements and Functions
FileCreateDataFile

FileEditDataFile

FileEditDataFile
Corresponds to clicking the Edit Data File button in the print merge bar; opens a merge document's associated data file.

_____

See also
Merge Statements and Functions

FileExit

FileExit [Save]
Corresponds to the Exit command on the File menu; quits Word.

Save	**Determines whether or not a save is forced:**
0 or omitted	The user is prompted to save each changed document.
1	All edited documents are automatically saved before exiting.
2	The documents are not saved.

_____

See also
Environment Statements and Functions
File Statements and Functions

FileFind

FileFind [.Title = text,] [.Subject = text,] [.Author = text,] [.Keywords = text,] [.SearchPath = text,] [.Text = text,] [.SavedBy = text,] [.DateCreatedFrom = text,] [.DateCreatedTo = text,] [.DateSavedFrom = text,] [.DateSavedTo = text,] [.Name = text,] [.Location = text,] [.MatchCase = number,] [.Options = number,] [.SortBy = number,] [.View = number,] [.SelectedFile = number]

Corresponds to the Find File dialog box (File menu); creates lists of files based on several search criteria. Can be used to change the search criteria in subsequent FileFind instructions. If you record a macro with FileFind, any other actions you perform in the Find File dialog box at that time (for example, opening or deleting a document, editing summary information, or printing) are recorded as separate instructions.

.Title	Title.
.Subject	Subject.
.Author	Author.
.Keywords	Keywords used to identify the document.
.SearchPath	A list of directories in which to search for files. Separate the directories with semicolons.
.Text	Text in the document. When multiple words are given for .Title, .Subject, .Author, .Keywords, .SearchPath, and .Text, they constitute a boolean search expression, rather than an English phrase, where spaces and commas are OR's and "&" is AND. For more information, see Special characters you can use in a document search.
.SavedBy	Name of the person who last saved the document.
.DateCreatedFrom	Document creation date you want to search from.
.DateCreatedTo	Document creation date you want to search to.
.DateSavedFrom	Document save date you want to search from.
.DateSavedTo	Document save date you want to search to.
.Name	Filename of the document; usually a file specification with a wildcard, such as "*.doc," "*.*," or "*.bmp," although an explicit filename could be used.
.Location	The path of the directory to search. Valid strings are as follows: **Path Only** — Search according to the .SearchPath. **All Local Drives** — Search on all local fixed drives. **All Drives** — Search on all local fixed and network drives. **X:** — Search all of drive X, where X is a drive letter.

.MatchCase	**Specifies whether or not to match the case of the search text exactly (applies only to the .Text argument):**
0 (zero)	Do not match the case (default).
1	Match the case.

.Options	**Where to place found files:**
0	Create new list.
1	Add matches to existing file list.
2	Search only in existing file list.

.SortBy	**Specifies how the documents are sorted:**
0	Alphabetically by author.
1	By creation date, with newest file listed first.
2	Alphabetically by the name of the person who last saved the document.
3	By date last saved, with newest file listed first
4	Alphabetically by filename.
5	By size, with smallest file listed first.

.View	**Specifies what is viewed in the right side of the dialog box:**
0	Displays title and sort criteria (if other than filename).
1	Displays the file's contents.
2	Displays summary information for the file.
3	Displays statistical information for the file.

.SelectedFile — Used only as part of a dialog box definition: Returns the index of the file selected in the files list in the Find File dialog box. To retrieve the filename, pass this value to the FoundFileName$() function (for example, FoundFileName(dlg.SelectedFile)).
Using this argument permits writing macros that let the user choose a file and then open, print, or perform some other operation on the file.

Example

```
FileFind .Name = "test*.doc", .Location = "Path Only", .Matchcase = 0,\
.Options = 0, .SortBy = 4, .View = 1
```

_____

See also

File Statements and Functions

FileName$()

a$ = FileName$([n])
Returns the name of the active document or a recently opened file as listed on the File menu.

n	The number of the file in the order listed on the File menu, from 1 through 4. If 0 (zero) or omitted, returns the name of the active document; if there is no active document, returns an empty string. If n is greater than the number of files listed on the File menu, an error is generated.

_____

See also
File Statements and Functions
CountFiles

FileNew

FileNew [.NewTemplate = number,] [.Template = text]
Corresponds to the New dialog box (File menu); opens an empty document window.

.NewTemplate	**Specifies whether to create a new document or a new template.**
0	Opens a new document window.
1 or omitted	Creates a new template.

.Template	Name of the document template or document on which to base the document or template.

Example
FileNew .NewTemplate = 1
Opens a new template called Template1.

_____
See also
File Statements and Functions

FileNewDefault

FileNewDefault
Creates a new document based on NORMAL.DOT.

_____
See also
File Statements and Functions

FileOpen

FileOpen .Name = text [,.ReadOnly = number] [,.PasswordDoc = text] [,.PasswordDot = text]
Corresponds to the Open dialog box (File menu); opens the specified document.

.Name	The name of the document.
.ReadOnly	If 1, the document is opened as read-only.
.PasswordDoc	The document's password.
.PasswordDot	The associated template's password.

An error is generated if the document does not exist.

Example
FileOpen .Name = "MYDOC.DOC", .ReadOnly = 1

_____
See also
File Statements and Functions

FileOpenDataFile

FileOpenDataFile .Name = text [,.ReadOnly = number] [,.PasswordDoc = text] [,.PasswordDot = text]
Opens a data file for print merge.

.Name	The full path of the data document to open.
.ReadOnly	If 1, the document is opened as read-only.
.PasswordDoc	The document's password.
.PasswordDot	The associated template's password.

Example

FileOpenDataFile .Name = "MYDATA.DOC"

_____

See also

Merge Statements and Functions

FileOpenHeaderFile

FileOpenHeaderFile .Name = text [,.ReadOnly = number] [,.PasswordDoc = text] [,.PasswordDot = text]

Opens a header file for print merge.

.Name	The full path of the data document to open.
.ReadOnly	If 1, the document is opened as read-only.
.PasswordDoc	The document's password.
.PasswordDot	The associated template's password.

Example

FileOpenHeaderFile .Name = "c:\WORDDOCS\MEADER.DOC"

_____

See also

Merge Statements and Functions

FilePrint

FilePrint [.Type = number,] [.NumCopies = text,] [.Range = number,] [.From = text,] [.To = text,] .PrintToFile = number, .Collate = number [,.FileName = text]
Corresponds to the Print dialog box (File menu); prints the active document.

.Type	Corresponds to the Print box; the type of document to print:
0	Document
1	Summary Information
2	Annotations
3	Styles
4	Glossary
5	Key Assignments

.NumCopies	Number of copies.

.Range	Page range:
0	Prints the entire document
1	Prints the selection. If the selection is an insertion point, prints the current page.
2	Prints the active page
3	Prints the specified range of pages

.From	Beginning of the range; can be page and/or section; ignored if Range equals 0 (zero) or 1.
.To	End of the range; can be page and/or section; ignored if Range equals 0 (zero) or 1.
.PrintToFile	Corresponds to the Print To File check box.
.Collate	Corresponds to the Collate Copies check box.
.FileName	If supplied, the specified file is printed. Recording Print from FileFind records FilePrint with this field.

_____

See also
File Statements and Functions

FilePrintDefault

FilePrintDefault
Prints the active document using the current defaults.

_____

See also
File Statements and Functions

FilePrintMerge

FilePrintMerge .Destination = number [,.MergeRecords = number] [,.From = text] [,.To = text] [,.Suppression = number] [,.SelectRecords]
Corresponds to clicking the Merge button in the Print Merge Setup dialog box (File menu); merges the data in a data document into a print merge document.

.Destination	Corresponds to the Merge Results option group; where to send the merged output:
0	To printer
1	To a new document
2	Only check errors

.MergeRecords	Specifies how to merge records:
0 or omitted	.From and .To are ignored; all records are merged.
1	Word merges only the records specified between From and To, one of which must be nonzero.

.From	The starting record number to merge.
.To	The ending record number to merge.

.Suppression	Corresponds to the Treatment Of Blank Lines option group:
0	Print blank lines
1	Skip completely
2	Move to bottom

.SelectRecords	Corresponds to clicking the Select Records button; brings up the Record Selection dialog box.

Example

```
FilePrintMerge .Destination = 1, .MergeRecords = 1, .From = 100,\
.To = 115, .Suppression = 0
```

Performs a print merge, using record numbers from 100 through 115, printing blank lines within merge fields.

_____

See also
File Statements and Functions

FilePrintMergeCheck

FilePrintMergeCheck .Destination = number [,.MergeRecords = number] [,.From = text] [,.To = text] [,.Suppression = number] [,.SelectRecords]
Corresponds to clicking in the print merge bar; checks for errors in a print merge document. Arguments are as for FilePrintMerge.

_____
See also
Merge Statements and Functions

FilePrintMergeReset

FilePrintMergeReset
Corresponds to clicking the Remove Attachments button in the Print Merge Setup dialog box (File menu); resets a print merge main document to a normal document.

_____
See also
Merge Statements and Functions

FilePrintMergeSelection

FilePrintMergeSelection [.MergeField1...MergeField6 = text,] [.ComparedTo1...ComparedTo6 = text,] [.CompOp1...CompOp6 = number]
Corresponds to clicking the Record Selection button in the Print Merge dialog box; specifies a set of records to merge into a merge document.

.MergeFieldN	The field to compare, which can be any of the merge fields used in the merge document, or the field Record Number.
.ComparedToN	The field, number, or string to compare with MergeFieldN.

.CompOpN	**The type of comparison to make, as listed in the Is list box in the Record Selection dialog box:**
1	Equal To
2	Not Equal To
3	Less Than
4	Greater Than
5	Lesser Than or Equal To
6	Greater Than or Equal To
7	Blank
8	Not Blank

Each argument ending in the same number specifies one complete condition for selecting records from the data document.

Example

```
FilePrintMergeSelection .MergeField1 = "Record Number", \
        .ComparedTo1 = 3, .CompOp1 = 6 \
        .MergeField2 = "Record Number", \
        .ComparedTo2 = 100, .CompOp2 = 5
```

Selects all records between record 3 and 100 inclusive.

_____
See also
Merge Statements and Functions

FilePrintMergeSetup

FilePrintMergeSetup [.RemoveAttachments,] [.DataFile,] [.HeaderFile,] [.MainDoc,] [.Merge]
Corresponds to clicking a button in the Print Merge Setup dialog box (File menu); prepares a main document for a print merge.

.RemoveAttachments	Corresponds to the Remove Attachments button.
.DataFile	Corresponds to the Attach Data File button.
.HeaderFile	Corresponds to the Attach Header File button.
.MainDoc	Corresponds to the Edit Main Doc button.
.Merge	Corresponds to the Merge button.

Example

```
FilePrintMergeSetup .Merge
```

Displays the Print Merge dialog box.

_____
See also
Merge Statements and Functions

FilePrintMergeToDoc

FilePrintMergeToDoc .Destination = number [,.MergeRecords = number] [,.From = text] [,.To = text] [,.Suppression = number] [,.SelectRecords]
Corresponds to clicking in the print merge bar; creates a new document with one form letter per section.
Arguments are as for FilePrintMerge.

_____
See also

FilePrintMergeToPrinter

FilePrintMergeToPrinter .Destination = number [,.MergeRecords = number] [,.From = text] [,.To = text] [,.Suppression = number] [,.SelectRecords]
Corresponds to clicking in the print merge bar; sends a series of print-merged documents to the printer.
Arguments are as for FilePrintMerge.

_____
See also
Merge Statements and Functions

FilePrintPreview

FilePrintPreview [On]
logical = FilePrintPreview()
Corresponds to the Print Preview command on the File menu; brings up or dismisses the Print Preview window.

On	Displays or removes the Print Preview window:
omitted	Toggles print preview.
1	Turns on print preview.
0	Turns off print preview.

The function form returns:

-1	If print preview is on.
0	If print preview is off.

_____

See also
File Statements and Functions
View Statements and Functions

FilePrintPreviewMargins

FilePrintPreviewMargins [On]
logical = FilePrintPreviewMargins()
Displays or removes text margins in print preview.

On	Displays or removes text margins:
1	Displays text margins.
0	Turns off display of text margins.
omitted	Toggles display of text margins.

The function form returns:

-1	If text margins are displayed.
0	If text margins are not displayed.

_____

See also
File Statements and Functions
View Statements and Functions

FilePrintPreviewPages

FilePrintPreviewPages [Pages]
logical = FilePrintPreviewPages()
Changes display in print preview between one and two pages.

Pages	Sets the number of pages to display:
0 or omitted	Toggles the display state (default).
1	Displays one page.
2	Displays two pages.

The function form returns:

-1	If one page is displayed.
0	If two pages are displayed.

_____

See also
File Statements and Functions
View Statements and Functions

FilePrintSetup

FilePrintSetup [.Printer = text,] [.Setup]
Corresponds to the Printer Setup dialog box (File menu); changes printing options for the active document.

.Printer	The name of the new printer to be activated. Enter this argument exactly as it appears in the Printer Setup dialog box.
.Setup	Displays a dialog box showing the printer options.

Example

FilePrintSetup "PostScript printer on COM2:"
Changes the printer to the PostScript printer attached to the COM2 port.

_____

See also
File Statements and Functions

Files$()

a$ = Files$(FileSpec$)
Returns the first filename that matches a file specification.

FileSpec$	**The file specification:**
omitted	The next file that matches the last-used FileSpec$ filename is returned.
	By specifying FileSpec$ on the first iteration and omitting it thereafter, you can use this function to get a list of files that match FileSpec$. If no files match, a null string (" ") is returned. Files$ (".") returns the current directory.

Examples
a$ = Files$("WIN.*")
Returns WIN.DOC, WIN.COM, WIN.TXT, or any file containing the text WIN.
CurrDir$ = Files$(".")
Returns the path of the current directory (for example, "C:\WINWORD").

_____
See also
File Statements and Functions
FileFind

FileSave

FileSave
Corresponds to the Save command on the File menu; saves the active document. If the document is not named, displays the Save As dialog box and prompts the user for the filename.

_____
See also
File Statements and Functions

FileSaveAll

FileSaveAll [Save]
Corresponds to the Save All command on the File menu; saves all changed files, including NORMAL.DOT.

Save	**Specifies whether or not to prompt the user to save changes:**
0	User is prompted to save all files marked as "dirty" (that is, changes have been made since the last time the file was saved).
1	All edited documents are automatically saved.

_____
See also
File Statements and Functions

FileSaveAs

FileSaveAs [.Name = text,] [.Format = number,] [.LockAnnot = number,] [.Password = text]
Corresponds to the Save As dialog box (File menu); saves the active document with a new name and/or format. If a file of the same name already exists, a message appears asking if you want to replace the existing file.

.Name	Specifies the new name.

.Format	**Specifies the new format:**
0	Normal (Word format).
1	Document Template.
2	Text Only (extended characters saved in ANSI character set).
3	Text+Breaks (plain text with line breaks; extended characters saved in ANSI character set).
4	Text Only (PC-8) (extended characters saved in IBM PC character set).
5	Text+Breaks (PC-8) (text with line breaks; extended characters saved in IBM PC character set).
6	Rich Text Format (RTF).

.LockAnnot	Locks the document for annotations.
.Password	Sets a password for the document.
	You can specify other file formats, which must be listed in the WIN.INI file under the Microsoft Word entry. The numbers for other formats begin at 100. The number for a format is its CONVNUM number in the INI file plus 100 minus 1.

Example
FileSaveAs .Name = "TEST.RTF", .Format = 6

Saves the active document under the name TEST.RTF, in RTF format.

_____

See also

File Statements and Functions

FileSummaryInfo

FileSummaryInfo [.Title = text,] [.Subject = text,] [.Author = text,] [.Keywords = text,] [.Comments = text,] [.Filename = text,] [.Directory = text,] [.Template = text,] [.CreateDate = text,] [.LastSavedDate = text,] [.LastSavedBy = text,] [.RevisionNumber = number,] [.EditTime = text,] [.LastPrintedDate = text,] [.NumPages = number,] [.NumWords = number,] [.NumChars = number,] [.Update]
Corresponds to the Summary Info dialog box (File menu); sets the summary information and allows access to the Statistics dialog box. All the options in the Statistics dialog box are read-only, with the exception of .EditTime.

.Title	Title.
.Subject	Subject.
.Author	Author.
.Keywords	Keywords used to identify the document.
.Comments	Comments on the document.
.Filename	Makes changes to summary information for the specified filename.
.Directory	The document's directory location; read-only.
.Template	Document template; read-only.
.CreateDate	Creation date; read-only.
.LastSavedDate	Date the document was last saved; read-only.
.LastSavedBy	Name of the person last saving the document; read-only.
.RevisionNumber	Number of times the document has been saved; read-only.
.EditTime	Total time the document has been open, in minutes.
.LastPrintedDate	Date document was last printed; read-only.
.NumPages	Number of pages; read-only.
.NumWords	Number of words; read-only.
.NumChars	Number of characters; read-only.
.Update	Corresponds to clicking the Update button; updates the summary information.

Example
FileSummaryInfo .Title = "Exploration of the Upper Amazon."
Sets the title of the active document to the specified text.

_____
See also
File Statements and Functions
DocumentStatistics

FileTemplate

FileTemplate .Store = number, .Template = text
Corresponds to the Template dialog box (File menu); changes the template and sets template options for the active document.

.Store	**Corresponds to the Store New Macros and Glossaries As option group:**
0	Global.
1	Document Template.
2 or omitted	Prompt when created.

.Template The full name of the template to associate with the active document.

Example
FileTemplate .Store = 0
Associates new macros and glossary entries with NORMAL.DOT.

_____
See also
File Statements and Functions

Font

Font Name$ [,Size]
a$ = Font$()
a$ = Font$(Count)
Applies a specified font to the selection.

Name$	The name of the font to apply.
Size	The size of the font, in points. You can use this argument instead of following Font with the FontSize instruction.
	The function form returns the font name of the selection. If the selection contains more than one font, a null string is returned. If Count is supplied, Font$() returns the name of the font Count, in the range 1 to CountFonts().

Example

Font "Courier", 8
Changes the selected text to 8-point Courier.

_____
See also
Formatting Statements and Functions

FontSize

FontSize Size
n = FontSize()
Sets the size of the selection, in points.
The function form returns the font size of the selection. If the selection has more than one font size, 0 (zero) is returned.

_____
See also
Formatting Statements and Functions

FootnoteOptions

FootnoteOptions [.FootnotesAt = number,] [.StartingNum = number,] [.RestartNum = number,] [.Separator,] [.ContSeparator,] [.ContNotice]
Corresponds to clicking the Options button in the Footnote dialog box and entering values in the Footnote Options dialog box; sets the placement and formatting of footnotes.

.FootnotesAt	**Where to place footnotes:**
0	End Of Section
1	Bottom Of Page
2	Beneath Text
3	End Of Document

.StartingNum	The starting number for footnotes in the active section.
.RestartNum	If nonzero, restarts footnotes in each section.
.Separator	Corresponds to clicking the Separator button; brings up the footnote separator pane.
.ContSeparator	Corresponds to clicking the Cont. Separator button; brings up the footnote continued separator pane.
.ContNotice	Corresponds to clicking the Cont. Notice button; brings up the footnote continuation notice pane.

Examples
FootnoteOptions .FootnotesAt = 3
Places footnotes at the end of the document.
FootnoteOptions .ContNotice
Displays the footnote continuation notice pane for the active document.

_____
See also
Formatting Statements and Functions

FormatBorder

FormatBorder [.FromText = text,] [.ApplyTo = number,] [.Shadow = number,] [.TopBorder = number,] [.LeftBorder = number,] [.BottomBorder = number,] [.RightBorder = number,] [.HorizBorder = number,] [.VertBorder = number,] [.TopColor = number,] [.LeftColor = number,] [.BottomColor = number,] [.RightColor = number,] [.HorizColor = number,] [.VertColor = number,] [.Pattern = number,] [.Foreground = number,] [.Background = number]
Corresponds to the Border dialog box (Format menu); sets border and shading formats for the selected paragraphs, picture, or table cells.

.FromText	The distance of the border from adjacent text, in points, or a measurement in the form of text. Valid only for paragraphs; otherwise, .FromText should be "" or omitted.

.ApplyTo	**If the selection consists of more than one of the following, specifies to what the border format is applied; if omitted, the default for the selection is assumed.**
0	Paragraphs
1	Picture
2	Cells
3	Whole table

.Shadow	**Specifies whether or not to apply a shadow to the border of paragraphs and pictures:**
0	None
1	Apply a shadow

.TopBorder, .LeftBorder, .BottomBorder, .RightBorder, .HorizBorder, .VertBorder
Specifies the border on the top, left, bottom, and right edges of paragraphs or cells, or between paragraphs and cells, in the range 0 (None) through 9.

.TopColor, .LeftColor, .BottomColor, .RightColor, .HorizColor, .VertColor
The color to be applied to the specified borders, in the range from 0 (Auto) through 16.

.Pattern The shading pattern to be applied to the selection, in the range from 0 (Auto) through 25.

.Foreground The color to be applied to the foreground of the shading, in the range from 0 (Auto) through 16.

.Background The color to be applied to the background of the shading, in the range from 0 Auto) through 16.

Example

```
FormatBorder .FromText = "1 in", .ApplyTo = 0, .TopBorder = 2,\
.LeftBorder = 0, .BottomBorder = 2, .RightBorder = 0, .HorizBorder = 0,\
.TopColor = 0, .BottomColor = 0, .Pattern = 0, .Foreground = 0,\
.Background = 0
```

Creates black, double-line top and bottom borders for the selected paragraphs, spaced 1 point from text.

_____

See also

Formatting Statements and Functions

FormatCharacter

FormatCharacter [.Font = text,] [.Points = value,] [.Bold = number,] [.Italic = number,] [.Strikethrough = number,] [.Hidden = number,] [.SmallCaps = number,] [.AllCaps = number,] [.Underline = number,] [.Color = number,] [.Position = value,] [.Spacing = value,] [.UseAsDefault]
Corresponds to the Character dialog box (Format menu); applies character formatting to the selection.

.Font	Name of font.
.Points	Font size, in points.
.Bold	Corresponds to the Bold check box.
.Italic	Corresponds to the Italic check box.
.Strikethrough	Corresponds to the Strikethrough check box.
.Hidden	Corresponds to the Hidden check box.
.SmallCaps	Corresponds to the Small Caps check box.
.AllCaps	Corresponds to the All Caps check box.

.Underline	**Corresponds to the Underline check box:**
0	None
1	Single
2	Words Only
3	Double

.Color	Color of the text; for a list of colors, see CharColor.

.Position	**The character's position in points, or as text specifying the amount and unit of measurement, relative to the baseline:**
0	Normal
>0	Superscript by the specified distance.
<0	Subscript by the specified distance.

.Spacing	**Specifies the spacing between characters, in points, or a measurement in the form of text:**
0	Normal
>0	Expanded by the specified distance.
<0	Condensed by the specified distance.

.UseAsDefault	Corresponds to the Use As Default button; sets the character formats of the Normal style.

Example
FormatCharacter .Spacing = "2 pt"
Sets character spacing for the selected text to an extra 2 points between characters.

_____
See also
Formatting Statements and Functions

FormatColumns

FormatColumns [.Columns = text,] [.ColumnSpacing = text,] [.ColLine = number,] [.StartNewCol = number,] [.ApplyColsTo = number]
Corresponds to options set in the Columns dialog box (Format menu); sets the column width and space between columns in the active section.

.Columns	The number of columns to set.
.ColumnSpacing	The space between columns.
.ColLine	Corresponds to the Line Between check box.
.StartNewCol	Corresponds to the Start New Column check box; if nonzero, the section starts in a new column.

.ApplyColsTo	**Specifies what to apply the column format to:**
0	Active section
1	From insertion point forward
2	Selected sections
3	Selected text
4	Whole document

Example
FormatColumns .Columns = "2", .StartNewCol = 1
Formats the active section for two columns, to start in a new column.

_____
See also
Formatting Statements and Functions

FormatDefineStyleBorder

FormatDefineStyleBorder [.FromText = text,] [.ApplyTo = number,] [.Shadow = number,] [.TopBorder = number,] [.LeftBorder = number,] [.BottomBorder = number,] [.RightBorder = number,] [.HorizBorder = number,] [.VertBorder = number,] [.TopColor = number,] [.LeftColor = number,] [.BottomColor = number,] [.RightColor = number,] [.HorizColor = number,] [.VertColor = number,] [.Pattern = number,] [.Foreground = number,] [.Background = number]
Sets border and shading formats for the current style. The arguments specify options available in the Border dialog box (Format menu).

_____
See also
Formatting Statements and Functions
FormatBorder
FormatStyle

FormatDefineStyleChar

FormatDefineStyleChar [.Font = text,] [.Points = value,] [.Bold = number,] [.Italic = number,] [.Strikeout = number,] [.Hidden = number,] [.SmallCaps = number,] [.AllCaps = number,] [.Underline = number,] [.Color = number,] [.Position = value,] [.Spacing = value]
Sets character formats for the current style. The arguments specify options available in the Character dialog box (Format menu).

_____
See also
Formatting Statements and Functions
FormatCharacter
FormatStyle

FormatDefineStyleFrame

FormatDefineStyleFrame [.Wrap = number,] [.WidthRule = number,] [.FixedWidth = text,] [.HeightRule = number,] [.FixedHeight = text,] [.PositionHorz = text,] [.PositionHorzRel = number,] [.DistFromText = text,] [.PositionVert = text,] [.PositionVertRel = number,] [.DistVertFromText = text,] [.MoveWithText = number,] [.RemoveFrame]
Sets frame formats for the current style. The arguments specify options available in the Frame dialog box (Format menu).

_____
See also
Formatting Statements and Functions
FormatFrame
FormatStyle

FormatDefineStyleLang

FormatDefineStyleLang .Language = text
Sets language formats for the current style. The arguments specify options available in the Language dialog box (Format menu).

_____
See also
Formatting Statements and Functions
FormatLanguage
FormatStyle

FormatDefineStylePara

FormatDefineStylePara [.Alignment = number,] [.LeftIndent = text,] [.RightIndent = text,] [.FirstIndent = text,] [.Before = text,] [.After = text,] [.LineSpacing = text,] [.PageBreak = number,] [.KeepWithNext = number,] [.KeepTogether = number,] [.NoLineNum = number]
Sets paragraph formats for the current style. The arguments specify options available in the Paragraph dialog box (Format menu).

_____
See also
Formatting Statements and Functions
FormatParagraph
FormatStyle

FormatDefineStyleTabs

FormatDefineStyleTabs [.Position = text,] [.DefTabs = text,] [.Align = number,] [.Leader = number,] [.Set,] [.Clear,] [.ClearAll]
Defines the current style with the specified tab formats. The arguments specify options available in the Tabs dialog box (Format menu).

_____
See also

FormatFrame

FormatFrame [.Wrap = number,] [.WidthRule = number,] [.FixedWidth = text,] [.HeightRule = number,] [.FixedHeight = text,] [.PositionHorz = number,] [.PositionHorzRel = number,] [.DistFromText = text,] [.PositionVert = number,] [.PositionVertRel = number,] [.DistVertFromText = text,] [.MoveWithText = number,] [.RemoveFrame]
Corresponds to the Frame dialog box (Format menu); sets position formats for the selected paragraphs or cells in a table.

.Wrap	**Corresponds to the Text Wrapping group:**
0	Does not wrap text around the frame.
1	Wraps text around the frame.

.WidthRule	**How to specify width of the frame:**
0	Auto width of frame determined by paragraph width.
1	Exactly

.FixedWidth Width of the frame, if .WidthRule is 1.

.HeightRule	**How to specify height of the frame:**
0	Auto; height of frame determined by paragraph height.
1	At Least
2	Exactly

.FixedHeight Height of the frame, if .HeightRule is 1 or 2.

.PositionHorz Absolute distance in points. You can specify a different unit by enclosing the measurement in quotation marks. You can also specify the following as text arguments:

Left
Center
Right
Inside
Outside

.PositionHorzRel	**Specifies horizontal position relative to:**
0	Margin
1	Page
2	Column

.DistFromText Distance between the frame and the text to the right and/or left of the frame.

.PositionVert Absolute distance in points. You can specify a different unit by enclosing the measurement in quotation marks. You can also specify the following as text arguments:

Top
Center
Bottom

.PositionVertRel	**Specifies horizontal position relative to:**
0	Margin
1	Page
2	Paragraph

.DistVertFromText Distance between the frame and the text above and/or below it.

.MoveWithText Corresponds to the Move With Text check box.

.RemoveFrame Corresponds to clicking the Remove Frame button; removes the frame format from the selected text.

Example

```
InsertFrame
FormatFrame .PositionHorz = 0, .PositionHorzRel = 2, \
  .DistFromText = "0.13 in"
```

Inserts a frame and formats it as left-aligned, relative to the current column, with a 0.13-inch gap between the frame and surrounding text.

_____
See also
Formatting Statements and Functions

FormatLanguage

FormatLanguage .Language = text [,.UseAsDefault]
Corresponds to the Language dialog box (Format menu); sets the language attribute for the selected text.

.Language	**The name of the language, from the following list:**
For	Use
No proofing	0 (zero)
Brazilian Portuguese	Português (BR)
Danish	Dansk
Dutch	Nederlands
English (AUS)	English (AUS)
English (UK)	English (UK)
English (US)	English (US)
Finnish	Suomi
French	Français
French Canadian	Français canadien
German	Deutsch
Italian	Italiano
Norwegian Bokmal	Norsk Bokmål
Norwegian Nynorsk	Norsk Nynorsk
Portuguese	Português (POR)
Spanish	Español
Swedish	Svenska

.UseAsDefault Corresponds to the Use As Default button.

Example
FormatLanguage .Language = "Português (BR)", .UseAsDefault
Makes Brazilian Portuguese an attribute of the Normal style.

_____
See also
Formatting Statements and Functions
Language

FormatPageNumber

FormatPageNumber [.NumFormat = number,] [.NumRestart = value,] [.StartingNum = text]
Corresponds to clicking the Format button in the Page Numbers dialog box (Insert menu); determines the format of the page number used in the selected section.

.NumFormat	**The format for page numbers:**
0	1 2 3...
1	a b c...
2	A B C...
3	i ii iii...
4	I II III...

.NumRestart	**Determines whether or not a different starting number can be set.**
0	Setting .StartingNum has no effect; equivalent to selecting the Continue From Previous Section option button.
1	Numbering begins at the number set for .StartingNum.

.StartingNum The active section's starting page number. The starting number is ignored if .NumRestart is set to 0 (zero).

_____
See also
Formatting Statements and Functions

FormatPageSetup

FormatPageSetup [.AttributeControls = number,] [.ApplyPropsTo = number,] [.TopMargin = text,] [.BottomMargin = text,] [.LeftMargin = text,] [.RightMargin = text,] [.Gutter = text,] [.FacingPages = number,] [.PageWidth = text,] [.PageHeight = text,] [.Orientation = number,] [.FirstPage = number,] [.OtherPages = number,] [.UseAsDefault]
Corresponds to the Page Setup dialog box (Format menu); sets the page setup within sections.

.AttributeControls	**Specifies which set of page setup properties are being modified:**
0	Margins
1	Size And Orientation
2	Paper Source

.ApplyPropsTo	**Specifies what part of the document to apply the page setup properties:**
0	Current Section
1	This Point Forward
2	Selected Sections
3	Selected Text
4	Whole Document

.TopMargin	The distance between the top edge of the page and the top boundary of the body text.
.BottomMargin	The distance between the bottom edge of the page and the bottom boundary of the body text.
.LeftMargin	The distance between the left edge of the page and the left boundary of the body text.
.RightMargin	The distance between the right edge of the page and the right boundary of the body text.
.Gutter	The distance allowed for the gutter.
.FacingPages	Corresponds to the Facing Pages check box; if nonzero, sets the Facing Pages option.
.PageWidth	The width of the page.
.PageHeight	The height of the page.

.Orientation	**The orientation of the page:**
0	Portrait
1	Landscape

.FirstPage, .OtherPages	**Selects the method of printing for the first page and the other pages in the document:**
0	Default Tray; determined by the printer driver
1	Upper Tray
4	Manual Feed; often used to override the default tray for the first page
5	Envelope

Other values may be available and depend on your printer driver.

.UseAsDefault	Corresponds to clicking the Use As Default button; makes the current page setup properties the default for new documents.

Example

```
FormatPageSetup .AttributeControls = 0, .ApplyPropsTo = 4,\
.TopMargin = "1 in"
```

Sets the current top margin to 1 inch for the entire document.

_____
See also
Formatting Statements and Functions

FormatParagraph

FormatParagraph [.Alignment = number,] [.LeftIndent = text,] [.RightIndent = text,] [.FirstIndent = text,] [.Before = text,] [.After = text,] [.LineSpacing = text,] [.PageBreak = number,] [.KeepWithNext = number,] [.KeepTogether = number,] [.NoLineNum = number]
Corresponds to the Paragraph dialog box (Format menu); applies paragraph formatting to the selected paragraphs.

.Alignment	**Sets a paragraph alignment:**
0	Left
1	Centered
2	Right
3	Justified

.LeftIndent	The left indent.
.RightIndent	The right indent.
.FirstIndent	The first-line indent.
.Before	The space before the paragraph.
.After	The space after the paragraph.
.LineSpacing	The spacing for all lines within the paragraph. With a positive measurement, the line height adjusts as needed to fit the tallest character in the line but is never less than the measurement you specify. With a negative measurement, the line height remains fixed regardless of character size.
.PageBreak	Corresponds to the Page Break Before check box; if nonzero, inserts a page break before printing the paragraph.
.KeepWithNext	Corresponds to the Keep With Next check box; if nonzero, prevents a page break after the paragraph.
.KeepTogether	Corresponds to the Keep Lines Together check box; if nonzero, prevents page breaks within a paragraph.
.NoLineNum	Corresponds to the Suppress check box; if nonzero, turns off line numbering for the paragraph.

Example
FormatParagraph .Alignment = 3, .Before = "1 in", .After = "1 in"
Sets justified alignment and adds 1 inch of space above and below each paragraph in the selection.

_____
See also
Formatting Statements and Functions

FormatPicture

FormatPicture [.SetSize = number,] [.CropTop = value,] [.CropLeft = value,] [.CropBottom = value,] [.CropRight = value,] [.ScaleX = text,] [.ScaleY = text,] [.SizeX = value,] [.SizeY = value]
Corresponds to the Picture dialog box (Format menu); applies picture formatting properties.

.SetSize	**Determines what arguments are used to specify the size of the picture:**
0	ScaleX and ScaleY are used to format the picture.
1	SizeX and SizeY are used to format the picture.

.CropTop, .CropLeft, .CropBottom, .CropRight	Amount to crop the picture, in points, or a measurement in the form of text. If a negative value is used, the picture is not cropped, but the amount of white space around the picture is increased.
.ScaleX, .ScaleY	Amount to scale the picture, as a percentage.
.SizeX, .SizeY	The horizontal and vertical dimensions of the picture, in points, or a measurement in the form of text.

Example
FormatPicture .ScaleX = "75%", .ScaleY = "75%"
Scales the selected picture to 75%.

_____
See also
Formatting Statements and Functions

FormatSectionLayout

FormatSectionLayout [.SectionStart = number,] [.VertAlign = number,] [.Footnotes = number,] [.LineNum = number,] [.StartingNum = number,] [.FromText = text,] [.CountBy = text,] [.NumMode]
Corresponds to the Section Layout dialog box (Format menu); applies section formatting properties to the selection.

.SectionStart	**Determines the type of section break:**
0	Continuous
1	New Column
2	New Page
3	Even Page
4	Odd Page

.VertAlign	**Alignment of section on the page:**
0	Top
1	Center
2	Justified

.Footnotes	Corresponds to the Suppress Footnotes check box.
.LineNum	Corresponds to the Add Line Numbering check box.
.StartingNum	Number at which to begin line numbering.
.FromText	Distance from text, in points, or a measurement in the form of text; a value of 0 sets automatic spacing.
.CountBy	Numerical increment used to print line numbers.

.NumMode	**Determines how lines are numbered:**
0	Every New Page
1	Every New Section
2	Continue

Example

FormatSectionLayout .SectionStart = 2, .VertAlign = 1
Formats the active section so that it starts on a new page, with centered alignment.

_____
See also
Formatting Statements and Functions

FormatStyle

FormatStyle .Name = text [,.KeyCode = number] [,.Apply] [,.BasedOn = text] [,.NextStyle = text] [,.AddToTemplate = number] [,.Define] [,.Delete] [,.Rename] [,.Merge] [,.NewName = text] [,.FileName = text] [,.Source = number]
Corresponds to the Style dialog box (Format menu); defines a style with the specified name, or applies the specified style. If a style with that name already exists, that style is made the current style.
FormatStyle sets up the style.
To define character formats, use FormatDefineStyleChar and FormatDefineStyleLang.
To define paragraph formats, use FormatDefineStylePara, FormatDefineStyleTabs, FormatDefineStyleBorder, and FormatDefineStyleFrame.
To redefine an existing style, include the specific arguments with the FormatStyle instruction.

.Name	The name of the style.
.KeyCode	A number representing the shortcut key sequence. For a table of keys and their values, see ToolsOptionsKeyboard.
.Apply	Applies the style to the selected paragraphs.
.BasedOn	Specifies a style on which to base the style.
.NextStyle	Specifies the style to be applied after the style.

.AddToTemplate	**Adds the style to the active template:**
0	The document only.
1	The document and its template.

.Define	Corresponds to the Add/Change button.
.Delete	Corresponds to the Delete button.
.Rename	Corresponds to the Rename button.
.Merge	Corresponds to the Merge button.
.NewName	Used only with the .Rename argument; specifies a new name for the style.
.FileName	Used only with the .Merge argument; specifies the template or document whose style sheet is to be merged with that of the active document or template.

.Source	**Used only in conjunction with the .Merge argument; specifies the source of the styles to be merged:**
0	From the active document or template to a specified document or template.
1	From a specified document or template to the active document or template. If no filename and argument are specified, the template of the active document is assumed.

Examples

FormatStyle .Name = "Title", .Define
Defines the new style Title.
FormatDefineStyleChar .Bold = 1
Redefines the current style (Title) to include bold formatting.
FormatDefineStylePara .Alignment = 2
Redefines the current style (Title) to include centered formatting.
FormatStyle .Name = "Headline", .Delete
Deletes the style Headline.
FormatStyle .Name = "Normal", .Keycode = 858, .Define
Establishes the keycode 858 (CTRL+SHIFT+Z) for the Normal style.

_____
See also
Formatting Statements and Functions

FormatTabs

FormatTabs [.Position = text,] [.DefTabs = text,] [.Align = number,] [.Leader = number,] [.Set,] [.Clear,] [.ClearAll]
Corresponds to the Tabs dialog box (Format menu); sets tab stops for the selected paragraphs.

.Position	Position of the tab stop, in points, or a measurement in the form of text.
.DefTabs	Position for default tab stops in the document, in points, or a measurement in the form of text.

.Align	**Alignment of the tab stop:**
0	Left
1	Centered
2	Right
3	Decimal

.Leader	**The leader character for the tab stop:**
0	None
1	Dot
2	Dash
3	Underline

.Set	Sets the specified tab stop.
.Clear	Clears the specified tab stop.
.ClearAll	Clears all tab stops.

Examples
Recording a set of tab actions generates a series of new commands, as shown in the following example:
FormatTabs .ClearAll
Clears all tabs.
FormatTabs .Position = "1.5 in", .Align = 2, .Set
Sets a right-aligned tab at 1.5 inches.
FormatTabs .Position = "3 in", .Clear
Clears the tab at 3 inches.
FormatTabs .Position = "4.5 in"
Positions a tab at 4.5 inches.

_____
See also
Formatting Statements and Functions

For...Next

For CounterVariable = Start To End [Step Increment]
...instruction(s)
Next [CounterVariable]
Executes the instructions between For and Next as many times as it takes CounterVariable to go from the Start value to the End value. Increment is the value used to increment the counter (the default is 1).

Note: When the For...Next loop is complete, CounterVariable = End + 1.

The CounterVariable with Next is optional. If CounterVariable is omitted, Next causes WordBasic to loop back to the first incomplete For instruction. Use of CounterVariable does slow program operation somewhat, and this can be important in longer and more complex programs. Using CounterVariable aids the readability of the program, but the same result can be obtained by adding a comment after the Next instruction (see Rem). Including CounterVariable does help check program logic for errors: You can use CounterVariable when writing and testing your programs and then delete it when the program has been finished and tested.

Examples
For count = 1 To 5
Next count
Value of count is incremented from 1 through 5 in steps of 1: 1, 2, 3, 4, 5.
For count = 1 To 3
  Beep
Next
The command Beep is repeated three times, as the value of count is incremented from 1 through 3 in steps of 1: 1, 2, 3.
For n = 5 To 1 Step -1
Next
Value of n is decremented from 5 through 1 in steps of -1: 5, 4, 3, 2, 1.
For n = 1 To 2 Step .2
Next
Value of n is incremented from 1 to 2 in steps of .2: 1, 1.2, 1.4, 1.6, 1.8, 2.

_____
See also

FoundFileName$()

a$ = FoundFileName$(n)
Returns the name of a found file. n is a number between 1 and CountFoundFiles().

_____
See also
File Statements and Functions
CountFoundFiles()
FileFind

Function...End Function

Function Name[(ParameterList)]
...instruction(s)
End Function
Defines a function. A function returns a single value. If a function should return a string value, the function name must end with a $. ParameterList is a list of variables, separated by commas, for receiving arguments to the function. String variables must end with the $ character. ParameterList cannot include values. Constants should be declared as variables and passed to the function by variable name.

Example

```
Sub MAIN
  StartOfDocument                    'Move insertion to start of doc

Mark:                                    'Beginning of loop
  If StartDot(1) Then                'Call StartDot with argument of 1
    EndOfLine 1                          'Select to end of line
    MsgBox"Period found at start of " + Chr$(34)\
      + Selection$() + Chr$(34)
                                         'Move insertion point back to start of line
    StartOfLine
  End If

                                         'If not at bottom of document, repeat
  If LineDown() = -1 Then Goto Mark
End Sub

Function StartDot(Flag)
  'Tests for period at beginning of current selection or in
  'character to right of current selection point. If found,
  '1 is returned. If argument of 1 is passed to
  'StartDot, beep sounds when period is found.
  If Left$(Selection$(),1) = "." Then
    StartDot = 1
    If Flag = 1 Then Beep
  End If
End Function
```

_____
See also
Standard Basic Statements and Functions

GetBookmark$()

a$ = GetBookmark$(BookmarkName$)
Returns the unformatted text marked by the specified bookmark. Operation is similar to Selection$().

_____
See also
Bookmark Statements and Functions

GetCurValues

GetCurValues DialogRecord
Stores in DialogRecord the current values for a previously dimensioned dialog box.
For more information, see "Using Macros" in the Microsoft Word User's Guide. For an example using GetCurValues, see DocumentStatistics.

_____
See also
Dialog Box Statements and Functions
Dialog

GetGlossary$()

a$ = GetGlossary$(Name$ [,Context])
Returns the text without formatting of the specified glossary entry.

Name$ A name of a glossary entry.

Context	**Scope of application:**
0 or omitted	Global
1	Document template

Example

```
new$ = GetGlossary$("Disclaim", 1)
Print new$
```

Sets new$ to the text in the document template glossary specified by the name Disclaim and displays the text in the status bar.

_____

See also

Glossary Statements and Functions

GetProfileString$()

a$ = GetProfileString$([App$,] Key$)
Gets a value from the current WIN.INI file.

App$	The name of the application. If the application is not specified, the string "Microsoft Word" is used.
Key$	The name of the WIN.INI option. If Key$ is not found, the function returns a null string.

_____

See also
Environment Statements and Functions

GetToolButton()

n = GetToolButton(Tool)
Returns an index to the button associated with the indicated tool, the button being the symbol or icon that is displayed. Corresponds to a position in the Button box in the Toolbar category of the Options dialog box (Tools menu).

Tool	Index to a tool on the Toolbar (0 being the leftmost tool)

_____

See also
Tools Statements and Functions

GetToolMacro$()

GetToolMacro$(Tool)
Returns the name of the macro associated with the specified tool. For example, GetToolMacro$(0) when called on an unmodified Toolbar returns "FileNew".

Tool	Index to a tool on the Toolbar (0 is the leftmost tool)

_____

See also
Tools Statements and Functions

GlossaryName$()

a$ = GlossaryName$(Count [,Context])
Returns the name of the glossary defined in the given context.

Count	The sequential number of the entry in the glossary, ranging from 1 through CountGlossaries(Context).

Context	**Scope of application:**
0 or omitted	Global
1	Document template

_____

See also
Glossary Statements and Functions

GoBack

GoBack
Corresponds to pressing SHIFT+F5; toggles among the last three selections where text or formatting has changed.

_____

See also
Selection Statements and Functions

Goto

Goto Label/LineNumber
Branches unconditionally to a label. A label can be a line number of 32,000 or less, or it can be a string that consists of an initial letter followed by letters or numbers up to a total length of 40 characters. Labels must start at the first character (position 0) of a line.

Example
...other instructions

```
If WordCount > 10000 Then Goto Novelette
...other instructions
Novelette:
...other instructions
```

_____

See also

Standard Basic Statements and Functions

GroupBox

GroupBox x, y, dx, dy, Text$
Creates a box with a label within a dialog box, often used to designate a group of option buttons or check boxes. A group box does not have a result.

x, y	The coordinates of the upper-left corner of the rectangle containing the group box and its associated label, in increments of 1/8th (for x) and 1/12th (for y) of the system font.
dx, dy	The width and height of the group box, in increments of 1/8th (for dx) and 1/12th (for dy) of the system font.
Text$	The associated text label displayed in the upper-left corner of the group box. An ampersand (&) preceding a character in Text$ makes that character the underlined access key for moving to the group box. For more information, see "Using Macros" in the Microsoft Word User's Guide.

Example
GroupBox 60, 29, 32, 37, "Status"

_____
See also
Dialog Box Statements and Functions

GrowFont

GrowFont
Corresponds to pressing CTRL+F2 in a document; increases the size of the selected font to the next available size supported by the selected printer. Can be used either on the selection or at the insertion point. If more than one font size is included in the selection, each is increased to its next available size.

_____
See also
Formatting Statements and Functions

HangingIndent

HangingIndent
Corresponds to pressing CTRL+T in a document; sets the left indent of the selection to the next tab stop in the first paragraph. Keeps the first line of the paragraph indented at the current left position.

_____
See also
Formatting Statements and Functions

Help

Help
Activates Help. Corresponds to pressing F1.

_____
See also
Help Statements and Functions

HelpAbout

HelpAbout
Displays a dialog box that gives the Word version number, serial number, registered user, available memory, disk space, presence of math coprocessor, and copyright information.

_____
See also
Help Statements and Functions

HelpActiveWindow

HelpActiveWindow
Corresponds to pressing F1 in the active window; activates Help for the active window.

_____
See also
Help Statements and Functions

HelpContext

HelpContext
Corresponds to pressing SHIFT+F1; activates context-sensitive Help, and changes the standard mouse pointer to a question mark.

_____
See also
Help Statements and Functions

HelpIndex

HelpIndex
Displays a list of Help topics.

_____
See also
Help Statements and Functions

HelpKeyboard

HelpKeyboard
Displays a list of keyboard Help topics.

_____
See also
Help Statements and Functions

HelpTutorialGstart

HelpTutorialGstart
Starts the Getting Started tutorial.

_____
See also
Help Statements and Functions

HelpTutorialLword

HelpTutorialLword
Starts the Learning Word tutorial.

_____
See also
Help Statements and Functions

HelpUsingHelp

HelpUsingHelp
Activates Help for using Help.

_____
See also
Help Statements and Functions

HelpWPHelp

HelpWPHelp
Shows equivalents for WordPerfect commands and key combinations.

_____
See also
Help Statements and Functions

Hidden

Hidden [On]
n = Hidden()
Corresponds to pressing CTRL+H; adds or removes the Hidden character format from the selected text. Word displays text having the Hidden format depending on whether you select Hidden Text or Show All, View options in the Options dialog box (Tools menu).

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format.
1	If all of the selection is in the format.

_____
See also
Formatting Statements and Functions

HLine

HLine [Count]
Scrolls the file contents horizontally, in lines. A "line" is the distance the display moves when the horizontal scroll bar arrow is clicked once.

Count	**The amount to scroll, in lines:**
omitted	One line to the right.
>0	Scrolls the screen to the right.
<0	Scrolls the screen to the left.

_____
See also
View Statements and Functions

HPage

HPage [Count]
Scrolls the file contents horizontally, in screen widths.

Count	The amount to scroll, in screen widths:
omitted	One screen to the right.
>0	Scrolls the screen to the right.
<0	Scrolls the screen to the left.

_____

See also
View Statements and Functions

HScroll

HScroll Percentage
n = HScroll()
Scrolls the file contents horizontally the specified percentage of the document width.The function form returns the current horizontal scroll position as a percentage of the document width.

_____
See also
View Statements and Functions

IconBarMode

IconBarMode
Corresponds to pressing SHIFT+F10. Places the cursor in the button bar, allowing a selection to be made using the arrow keys. The macro suspends operation and the mouse pointer becomes an hourglass until a choice is made. If a selection is not made, an error occurs.

_____
See also
Environment Statements and Functions

If...ElseIf...Else...End If

If Condition Then Statement(s) [Else Statement(s)]
If Condition1 Then
...instruction(s)
[ElseIf Condition2 Then
...instruction(s)]
[Else
...instruction(s)]
End If
Executes the specified instructions, depending whether the expression specified as Condition evaluates to zero (that is, false) or nonzero (that is, true).

Examples
If n = 10 Then Beep
Beeps if n equals 10.
If n = 10 Then Beep Else GoPrint
Beeps if n equals 10; otherwise, starts the GoPrint subroutine.

```
If n = 10 Then
  Beep
Else
  GoPrint
End If
```

Is equivalent to the preceding construction, but permits using more than one instruction in each group.

```
If n = 10 Then
  Beep
ElseIf name$ = "George" Then
  GoPrintGeorge
Else
  GoPrint
End If
```

Beeps if n equals 10; if not, executes the GoPrintGeorge routine if name$ equals "George" or starts the GoPrint subroutine. You can use several ElseIf clauses within an If...End If construction.

_____
See also
Standard Basic Statements and Functions

Indent

Indent
Indents the selected paragraphs; corresponds to pressing CTRL+N. The indent is aligned with the next tab stop of the first paragraph in the selection. Indent does not change the setting of a first-line indent.

_____
See also
Formatting Statements and Functions

Input

Input #StreamNumber, Variable [,Variable]
Input [Prompt$,] Variable [,Variable]
Each input instruction reads one line from the file specified by #StreamNumber into the variables listed. Use Open to open the file and establish a streamnumber for the file. The line read from the file is separated into individual values by commas. In the source file, precede a comma with a space to avoid truncating the final character of each value. Spaces following a comma in the source file are input as a leading space for the value following. Because a maximum of 256 characters can be read in, an error will be generated if this instruction is used on Word document or template files.
If a StreamNumber is not specified, the user is prompted in the status bar for keyboard input. The prompt ? is always displayed. An optional prompt string can also be supplied.

Examples
Input#1, Name$, ZipCode

Reads a line from stream #1 into the variables Name$ and ZipCode.
Input "File Search Spec", Spec$
Displays the prompt "File Search Spec?" in the status bar and loads the resulting keyboard input into the variable Spec$.

_____
See also
File I/O Statements and Functions
Open...For...As

Input$()

a$ = Input$(n, StreamNumber)
Reads n characters (up to 32,767) from the file specified by StreamNumber. Use Open to open the file and establish a streamnumber for the file. An error will be generated if this instruction is used on Word document or template files.

_____
See also
File I/O Statements and Functions
Open

InputBox$()

a$ = InputBox$(Prompt$ [,Title$] [,Default$])
Displays a dialog box requesting a single item of data, and returns the text entered in the dialog box when the user clicks OK.

Prompt$	The text displayed in the dialog box.
Title$	The title displayed in the title bar of the dialog box; if omitted, Word uses the title "Microsoft Word".
Default$	The default text proposed in the text box of the dialog box.

Example
FilePrn$ = InputBox$("File to print?","My Print Macro", "LTR.DOC")
Asks for the name of a file to print, defaulting to the document LTR.DOC.
num = Val(InputBox$("Input a number from 1 to 10"))
The Val() function returns the numeric value, if any, of a string. If in response to the prompt created by this instruction the user typed 10, then Val() would assign the value 10 to num. If, however, the user typed Ten, Val() would assign the value 0 to num.

_____
See also
Dialog Box Statements and Functions
Val()

Insert

Insert Text$
Inserts the specified text at the insertion point. You can include a nonprinting character by concatenating alphanumeric text with a Chr$() function.

Note: Do not insert paragraph marks by inserting Chr$(13) + Chr$(10); use InsertPara instead.

Examples
Insert "Hamlet"
Inserts the text Hamlet at the insertion point.
Insert Chr$(34) + "Hamlet" + Chr$(34)
Inserts the text "Hamlet" at the insertion point.

_____
See also
Editing Statements and Functions
Chr$()
InsertPara

InsertAnnotation

InsertAnnotation
Inserts a comment and activates the annotations pane.

_____
See also
Editing Statements and Functions

InsertBookmark

InsertBookmark .Name = text [,.Delete]
Corresponds to the Bookmark dialog box (Insert menu); creates or deletes the specified bookmark.

.Name	The name of the bookmark.
.Delete	Deletes the bookmark; if omitted, WordBasic creates the bookmark using the selected text.

Examples

InsertBookmark .Name = "begin"
Inserts the bookmark begin at the active selection or insertion point.
InsertBookmark .Name = "begin", .Delete
Deletes the bookmark begin.

_____
See also
Bookmark Statements and Functions

InsertBreak

InsertBreak .Type = number
Corresponds to the Break dialog box (Insert menu); inserts a page, section, or column break at the insertion point or selection.

Type	**The type of break:**
0	Page
1	Column
2	Next Page section break
3	Continuous section break
4	Even Page section break
5	Odd Page section break

_____
See also
Formatting Statements and Functions

InsertChart

InsertChart
Activates the Microsoft Graph application.

_____
See also
Editing Statements and Functions

InsertColumnBreak

InsertColumnBreak
Corresponds to pressing CTRL+SHIFT+ENTER; inserts a column break at the insertion point. If the insertion point is in a table, the break is inserted above the row in which the insertion point is located.

Equivalent to:
InsertBreak .Type = 1

_____
See also
Formatting Statements and Functions

InsertDateField

InsertDateField
Corresponds to pressing ALT+SHIFT+D; inserts a DATE field at the selection. If a header or footer pane is open, inserts the field in the header or footer.

_____
See also
Field Statements and Functions

InsertDateTime

InsertDateTime [.DateTimePic = text]
Corresponds to the Date and Time dialog boxes (Insert menu); inserts as a DATE field the current date and/or time into the active document.

.DateTimePic A string describing the format used for displaying the date and/or time; if omitted, uses the current date and time format according to the Control Panel's international settings.

Example
InsertDateTime .DateTimePic="d MMMM, yyyy"
Inserts the current date in the form 3 September, 1992.

_____
See also
Editing Statements and Functions

InsertDrawing

InsertDrawing
Activates the Microsoft Draw application.

_____
See also
Editing Statements and Functions

InsertField

InsertField .Field = text
Corresponds to the Field dialog box (Insert menu); inserts the specified field at the selection.

.Field The field to insert, as listed in the Field dialog box.

Do not include the field characters in Field$, but follow all other syntax rules for field codes. To insert quotation marks in field codes, use Chr$(34).

Example
InsertField .Field = "Author"
Inserts an AUTHOR field.

InsertField .Field = "createdate \@ " + Chr$(34) + "d MMMM, yyyy" + \ Chr$(34)
Inserts a field for the creation date of the document, in the form "1 July, 1992".

_____
See also
Field Statements and Functions

InsertFieldChars

InsertFieldChars
Inserts field characters ({}) at the selection; corresponds to pressing CTRL+F9.

_____
See also
Field Statements and Functions

InsertFile

InsertFile .Name = text [,.Range = text] [,.Link = number]
Corresponds to the File dialog box (Insert menu); inserts the specified file at the insertion point or selection.

.Name	The name of the file to insert.
.Range	If .Name refers to a Word document, .Range refers to a bookmark. If .Name refers to another document type (for example, a Microsoft Excel worksheet), then .Range refers to a named range or cell range (for example, R1C1:R3C4). Only that part of the file is inserted; to link the entire file, specify .Range = "".
.Link	If .Link is nonzero, an INCLUDE field is inserted instead of the text of the file itself.

Examples

InsertFile .Name = "PRICES.TXT"
Inserts the contents of the file PRICES.TXT into the active document.
InsertFile .Name = "PRICES.DOC", .Range = "sportscars"
Inserts the portion of PRICES.DOC associated with the bookmark "sportscars" into the active document.
InsertFile .Name = "PRICES.DOC", .Range = "sportscars", .Link = -1
Inserts the following field: {INCLUDE PRICES.DOC sportscars}
InsertFile .Name = "PRICES.DOC", .Range = "", .Link = -1
Inserts the following field: {INCLUDE PRICES.DOC}

_____
See also
Editing Statements and Functions
File Statements and Functions

InsertFootnote

InsertFootnote [.Reference = text]
Corresponds to the Footnote dialog box (Insert menu); inserts a footnote reference mark at the insertion point or selection and opens the footnote pane.

.Reference	The footnote reference marker that you supply. If no marker is supplied, an automatically numbered footnote reference is inserted.

Example

InsertFootnote .Reference = "*"
Inserts a footnote marked by an asterisk.

_____
See also
Editing Statements and Functions
FootnoteOptions

InsertFrame

InsertFrame
Inserts an empty frame, or frames the selected text. If no selection is made, Word inserts a 1-inch square frame at the insertion point. You can change the dimensions with FormatFrame.

_____
See also
Editing Statements and Functions
FormatFrame
RemoveFrames

InsertIndex

InsertIndex [.Type = number] [,.HeadingSeparator = number] [,.Replace = number]
Corresponds to the Index dialog box (Insert menu); inserts an INDEX field at the insertion point or selection.

.Type	**The type of index:**
0 or omitted	Normal Index
1	Run-in Index

.HeadingSeparator	**The heading separator:**
0 or omitted	None
1	Blank line
2	Letter

.Replace	**Specifies whether to replace an existing index or insert the new index at the insertion point:**

0 or omitted	User is prompted before an existing index is overwritten.
1	Existing index is overwritten.

Example

InsertIndex .Replace = 1

Creates a new index having no heading separator, without prompting the user to verify the replacement of an existing index.

_____

See also

Field Statements and Functions

InsertIndexEntry

InsertIndexEntry [.Entry = text] [,.Range = text] [,.Bold = number] [,.Italic = number]
Corresponds to the Index Entry dialog box (Insert menu); inserts an XE field at the insertion point or selection.

.Entry	The text of the index entry itself; if omitted, the selected text becomes the entry.
.Range	The name of a bookmark that specifies the range of text to which the index entry applies.
.Bold	Corresponds to the Bold check box.
.Italic	Corresponds to the Italic check box.

Example
InsertIndexEntry .Entry = "Using Tabs", .Bold = 1
Inserts the index entry "Using Tabs" at the insertion point, formatted in bold in the index.

_____
See also
Field Statements and Functions

InsertMergeField

InsertMergeField .MergeField = text, .WordField = number
Corresponds to clicking the Insert Merge Field button on the print merge bar of the active document; inserts a print merge field at the insertion point.

.MergeField	The name of the field to insert, from the associated data document.

.WordField	**The number of one of the Word merge fields, as listed in the Insert Merge Field dialog box.**
0	Ask
1	Fillin
2	If...Then
3	If...Then...Else
4	Merge Record #
5	Next Record
6	Next Record If
7	Quote
8	SetBookmark
9	Skip Record If

Example
InsertMergeField .MergeField = "", .WordField = 0
Inserts the ASK field at the insertion point or selection.

_____
See also
Merge Statements and Functions

InsertObject

InsertObject .Class = text
Corresponds to the Object dialog box (Insert menu); activates the server applications and inserts an EMBED field of the specified type.

.Class	The name of the class of object to be inserted.

Example
InsertObject "Word Document"
Results in the insertion of the following field in the document:
{EMBED WordDocument \s * mergeformat}

_____
See also
Editing Statements and Functions

InsertPageBreak

InsertPageBreak
Corresponds to pressing CTRL+ENTER; inserts a page break at the insertion point or selection.

_____
See also
Formatting Statements and Functions
InsertBreak

InsertPageField

InsertPageField
Corresponds to pressing ALT+SHIFT+P; inserts a PAGE field in the header or footer, if open; if not, inserts the page number at the insertion point or selection.

_____
See also
Field Statements and Functions
InsertDateField

InsertPageNumbers

InsertPageNumbers [.Type = number,] [.Position = number]
Corresponds to the Page Numbers dialog box (Insert menu); inserts a PAGE field into the header or footer. Prompts to replace the current header or footer with a simple page-number header or footer. Be aware that InsertPageNumbers, like the Page Numbers command, selects the Different First Page check box and clears the Different Odd And Even Pages check box in the Header/Footer dialog box (View menu).

.Type	The type of PAGE field to insert:
0	Top of page (Header)
1	Bottom of page (Footer)

.Position	The alignment of the page number:
0	Left
1	Center
2	Right

Example
InsertPageNumbers .Type = 0, .Position = 2
Inserts a page number at the top of the document, right-aligned.

_____
See also
Field Statements and Functions

InsertPara

InsertPara
Corresponds to pressing ENTER; inserts a paragraph mark at the insertion point or selection.

_____
See also
Formatting Statements and Functions

InsertPicture

InsertPicture .Name = text, .LinkToFile = number
Corresponds to the Picture dialog box (Insert menu); inserts a picture or an IMPORT field at the insertion point or selection.

.Name	The filename of the picture to import.

.LinkToFile	Corresponds to the Link To File check box:
0 or omitted	Inserts the picture specified by Name$.
1	Inserts an IMPORT field at the selection.

Example
InsertPicture .Name = "chess.bmp", .LinkToFile = 1

_____
See also
Field Statements and Functions

InsertSymbol

InsertSymbol .Font = text, .CharNum = text
Corresponds to the Symbol dialog box (Insert menu); inserts a character or a SYMBOL field at the insertion point.

.Font	The font of the symbol. If you specify an ANSI font -- for example, Times Roman -- Word inserts a character. If you specify Symbol or another decorative font, Word inserts a SYMBOL field.
.CharNum	The ANSI code of the symbol to insert.

Example
InsertSymbol .Font = "Symbol", .CharNum = 65
Inserts the following field:
{SYMBOL 65 \f "Symbol"}

_____
See also
Field Statements and Functions

InsertTableOfContents

InsertTableOfContents [.Source = number,] [.From = text,] [.To = text,] [.Replace = value]
Corresponds to the Table Of Contents dialog box (Insert menu); inserts a Table Of Contents field at the insertion point or selection.

.Source	Determines the source of the table of contents:
0 (zero)	Heading-style paragraphs
1	TC (Table of Contents Entry) fields

.From	The starting outline level used.
.To	The ending outline level used.

.Replace	Specifies whether to replace the current table of contents or insert the new table of contents at the insertion point or selection:
0 or omitted	User is prompted before an existing table of contents is overwritten.
1	The existing table of contents is overwritten.

Example

```
InsertTableOfContents .Source = 0, .From = "1", .To = "3", .Replace = 0
```

_____
See also
Field Statements and Functions

InsertTimeField

InsertTimeField
Corresponds to pressing ALT+SHIFT+T; inserts a TIME field at the insertion point or selection, but if the header/footer pane is open, inserts the field in the header or footer.

_____
See also
Field Statements and Functions

InStr()

n = InStr([Index,] Source$, Search$)
Searches for one text string in another.

Index	The starting character position of the search.
Source$	The text to be searched.
Search$	The text to search for.

The function returns the character position of the beginning of the text string, or zero if Search$ is not found in Source$.

Examples

```
Pos = InStr("Testing","ing")
```
Sets the variable Pos to the number 5.

```
Pos = InStr("Microsoft","o")
```
Sets the variable Pos to the number 5.

```
Pos = InStr(6,"Microsoft","o")
```
Sets the variable Pos to the number 7.

_____
See also
Standard Basic Statements and Functions

Int()

n = Int(n)
Returns the integer part of n.

Example

```
x = Int(98.6)
```
Sets x to the number 98.

```
x = Int(-9.6)
```
Sets x to the number -9.

_____
See also
Standard Basic Statements and Functions

IsDirty()

logical = IsDirty()

The function returns:

0 (zero)	If the document has not changed since it was last saved.
-1	If the document has been changed (made "dirty") since the last time it was saved

_____

See also
Environment Statements and Functions

IsExecuteOnly

logical = IsExecuteOnly([macro$])
Returns 0 (zero) if the specified macro can be edited or -1 if the macro is execute-only.

macro$	The name of a macro to examine, in the form [templateName:]macroName. If omitted, the active template and active macro are assumed.

Example

```
print IsExecuteOnly("NORMAL.DOT:TestMacro")
```

Displays -1 in the status bar if the macro is execute-only.

_____

See also
Environment Statements and Functions
MacroCopy

Italic

Italic [On]
n = Italic()
Corresponds to pressing CTRL+I; adds or removes the Italic character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0	Removes the format.
omitted	Toggles the format.

The function form returns:

0	If none of the selection is in the format.
-1	If part of the selection is in the format.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

JustifyPara

JustifyPara
n = JustifyPara()
Corresponds to pressing CTRL+J; justifies the selected paragraphs.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

KeyCode()

n = KeyCode(KeyList [,.Context = value])
Returns the keycode associated with the specified macro or built-in command, as it occurs in the current keycode list. The keycode list contains the key assignments in the Keyboard category of the Options dialog box (Tools menu) that differ from the default assignments.
For a list describing the convention for keycodes, see ToolsOptionsKeyboard.

.Context	**The context of the keycode list:**
0 or omitted	Global
1	Template

_____

See also
Environment Statements and Functions
CountKeys()

KeyMacro$()

a$ = KeyMacro$(KeyList [,.Context = value])

Returns the name of the macro or built-in command associated with a keycode, as it appears in the current keycode list. The keycode list contains the key assignments in the Keyboard category of the Options dialog box (Tools menu) that differ from the default assignments.
For a list describing the convention for keycodes, see ToolsOptionsKeyboard.

.Context	**The context of the keycode list:**
0 or omitted	Global
1	Template

See also
Macro Statements and Functions
CountKeys()

Kill

Kill Name$
Deletes the specified file.

Name$	The name of the file to delete; if the full path is not specified, the current directory is assumed.

Example

Kill "C:\WORD\LETTERS\DRAFT.DOC"

_____
See also
File Statements and Functions

Language

Language Language$
a$ = Language$()
a$ = Language$(Count)
Formats the selected text as a certain language. Word uses the proofing tools of the language you specify on this text.
The function form returns the language in which the first character of the selected text is formatted. If Count is supplied, Language$() returns the name of the language Count, in the range 1 through CountLanguages().

Examples
Language "Italiano"
Formats selected text as Italian. Word uses Italian proofing tools on this text.
Print Language$()
If the first character of the selected text is formatted in the English language, displays English (US) in the status bar.

_____
See also
Formatting Statements and Functions
FormatLanguage

LCase$()

a$ = LCase$(Source$)
Returns Source$ converted to lowercase.

_____
See also
Standard Basic Statements and Functions

Left$()

a$ = Left$(Source$, n)
Returns the leftmost n characters of Source$.

Examples
a$ = "Legal File List"
Print Left$(A$,5)
Returns the text Legal.
a$ = "Legal File List"
Print Left$(A$,10)
Returns the text Legal File.

_____
See also
Standard Basic Statements and Functions

LeftPara

LeftPara
n = LeftPara()
Corresponds to pressing CTRL+L; left-aligns the selected paragraphs.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of other alignments.
1	If all of the selection is in the format.

_____
See also
Formatting Statements and Functions

Len()

n = Len(Source$)
Returns the number of characters in Source$.

Example
a$ = "Trey Research"
Print Len(A$)
Displays the number 13 in the status bar.

_____
See also
Standard Basic Statements and Functions

Let

[Let] Var = Expression
Assigns the value of an expression to a variable. Let is optional and not commonly used.

Examples
The following assignment instructions are all valid:
Let A = 100
A = 100
Let Discount = Gross - (Gross * .10)
Discount = Gross - (Gross * .10)
Let StockNum$ = "AB4100-2"
StockNum$ = "AB4100-2"

_____
See also
Standard Basic Statements and Functions

Line Input

Line Input #StreamNumber, Variable$
Line Input [Prompt$] [,Variable$]
Reads an entire line (that is, all text until the next carriage return-line feed combination) from the file specified by StreamNumber and puts the result in the specified string variable. Use Open to open the file and establish a streamnumber for the file. Because a maximum of 256 characters can be read in, an error will be generated if this instruction is used on Word document or template files.
If StreamNumber is omitted, the user is prompted with a question mark (?) in the status bar for keyboard input. An optional prompt string can also be supplied. If Prompt$ is supplied, ? is not included.
Line Input is similar to the Input statement, but Line Input does not break the line into separate values at commas, placing its result in a single string variable, and does not include a question mark with a supplied prompt string.

Examples
Line Input #1, Sample$
Loads one line from the file attached to stream #1 into the variable Sample$.
Line Input Key$
Places the prompt ? in the status bar and places the resulting keyboard input into the string variable Key$.
Line Input "Search text:", Target$
Places the prompt Search text: in the status bar and loads the resulting keyboard input into the string variable Target$.
Line Input "", Code$
Accepts keyboard input in the status bar but displays no prompt. Loads the resulting keyboard input into the string variable Code$.

_____
See also
File Statements and Functions
Input
Open

LineDown

LineDown [Count,] [Select]
logical = LineDown([Count,] [Select])
Corresponds to pressing the DOWN ARROW key; moves the insertion point down or extends the selection by the specified number of lines.

Count	The number of lines to move down; if omitted, 1 is assumed.
Select	If nonzero, the selection is extended downward by Count lines.
	The function form returns 0 (zero) if the action cannot be completed.

Examples
LineDown 1, 1
Extends the selection down one line from the insertion point.
While LineDown(,1)
Wend
Extends the selection one line at a time until the end of the document is reached.

_____
See also
Selection Statements and Functions

LineUp

LineUp [Count,] [Select]

logical = LineUp([Count,] [Select])
Corresponds to pressing the UP ARROW key; moves the insertion point up or extends the selection upward by the specified number of lines.

Count	The number of lines to move up; if omitted, 1 is assumed.
Select	If nonzero, the selection is extended upward by Count lines.
	The function form returns 0 (zero) if the action cannot be completed.

Example
LineUp 20
Moves the insertion point up 20 lines from the previous insertion point.

_____
See also
Selection Statements and Functions

ListBox

ListBox x, y, dx, dy, ArrayVariable$(), .Field
Creates a list box at the specified location.

x, y	The coordinates of the upper-left corner of the list box, in units of 1/8th (for x) and 1/12th (for y) of the system font.
dx, dy	The width and height of the list box, in units of the system font.
ArrayVariable$()	A string array containing the list, one line per array element.
.Field	The user's selection from the list box, which can serve as an index in ArrayVariable$().
	For more information, see "Using Macros" in the Microsoft Word User's Guide.

Example
ListBox 6, 31, 49, 33, Arr$(), .Style
Creates a list box at (6, 31) which is 49 pixels wide and 33 pixels tall, filled with entries contained in the string array Arr$. The user's selection in the list is returned in .Style.

_____
See also
Dialog Box Statements and Functions

LockFields

LockFields
Corresponds to pressing CTRL+F11 (or ALT+SHIFT+F1); prevents the fields within the selection from being updated.

_____
See also
Field Statements and Functions

Lof()

n = Lof(StreamNumber)
Returns the length of the file, in bytes.

Example
```
Sub MAIN
  filename$ = "MYDATA.TXT"
  Open filename$ For Input As #1          'Set file for sequential input.
                                          'Connect to stream number one
    Size = Lof(1)                         'Put file size into variable "Size"
    Print Size                            'Display file size in status bar
  Close                                   'Close file
End Sub
```

_____
See also
File I/O Statements and Functions

MacroCopy

MacroCopy Macro1$, Macro2$ [, ExecuteOnly]
Copies a macro from a source to a destination, and optionally makes the macro execute-only; execute-only macros are encrypted and cannot be edited. Both templates must be open, and MacroCopy cannot replace an open macro.

Macro1$	The source or original macro from which to copy.
Macro2$	The destination or new macro.
ExecuteOnly	If nonzero, makes the destination macro execute-only.
	Macro names must be in the form [templateName:]macroName. For example, each of the following macro names are acceptable:

macroName
TemplateName:MacroName
Global:MacroName
c:\winword\normal.dot:test
If no template name is given, NORMAL.DOT is assumed.

Examples
MacroCopy "Macro1", "MyTemp:MyMacro1"
Copies Macro1 from the active template to the MyTemp template.
MacroCopy "Global:Macro1", "Macro2"
Copies Macro1 from the NORMAL.DOT template to the active template, and names the copy Macro2.

MacroCopy "Original", "ExecOnly", 1
In the active template, copies the Original macro to an execute-only macro named ExecOnly.

_____
See also
Macro Statements and Functions
IsExecuteOnly

MacroDesc$()

a$ = MacroDesc$(MacroName)
Returns the description string associated with the specified macro.

MacroName	The name of a macro, as text.

_____
See also
Macro Statements and Functions

MacroName$()

a$ = MacroName$(Count [,Context] [,All])
Returns the name of the macro defined in the specified context.

Count	The macro's number, taken from the list in the given context; it can range from 1 through CountMacros(Context, All).

Context	**Scope of the specified macro:**
0 (zero)	Global
1	Document template; using 1 for a macro in NORMAL.DOT causes an error.
All	If true, built-in commands are included and are numbered before macros you create.

MacroName$(0) gives the name of the active macro window, if any. The name of the active macro
window appears in the macro editing bar.

_____
See also
Macro Statements and Functions

MenuMacro$()

a$ = MenuMacro$(Menu, Item, [Context])
Returns the macro name associated with the specified menu.

Menu	The number of the specified menu: 1 for File, 2 for Edit, etc.
Item	The number of the menu command on the menu, from 1 through CountMenuItems().

Context	**Scope of the specified command:**
0 or omitted	Global
1	Document template

_____
See also
Macro Statements and Functions

MenuMode

MenuMode
Corresponds to pressing ALT or F10; activates menu mode.

_____
See also
Environment Statements and Functions

MenuText$()

a$ = MenuText$(Menu, Item, [Context])
Returns the name of the specified menu command.

Menu	The number of the specified menu: 1 for File, 2 for Edit, etc.
Item	The number of the menu command on the menu, from 1 through CountMenuItems().

Context	**Scope of the specified command:**

0 or omitted	Global
1	Document template

_____

See also
Macro Statements and Functions

MergeFieldName$()

a$ = MergeFieldNames$(Index)
Returns the field name corresponding to the specified index.

Index	The number of the field name in the header record of the data file or header file associated with the active merge document.

Example

```
For n = 1 to CountMergeFields()
  Insert MergeFieldName$(n)
  InsertPara
Next
```

Inserts a list of merge field names into the active document.

_____

See also
File Statements and Functions

Mid$()

n = Mid$(Source$, Index [,Count])
Returns Count characters from Source$, starting at character Index. If Count is not supplied, the rest of the string is returned.

Examples
Print Mid$("ABCDEFGH", 2, 3)
Returns the value BCD.
Print Mid$("ABCDEFGH", 2)
Returns the value BCDEFGH.

_____

See also
Standard Basic Statements and Functions

MkDir

MkDir Name$
Creates the directory specified by Name$.

Example
MkDir "E:\PROGRAMS\EXCEL"

_____

See also
File Statements and Functions

MoveText

MoveText
Moves text; equivalent to pressing F2. To use this statement, record or include in a macro instructions for the following series of steps:

1	Make a selection.
2	Run MoveText.
3	Position the insertion point where you want to move the text.
4	Run OK or press ENTER.

_____

See also
Editing Statements and Functions

MsgBox

MsgBox Message$ [,Title$] [,Type]
Displays a message box. For information on the function form, see MsgBox().

Message$	The message to appear in the message box.

Title$ The title of the message box. If omitted, "Microsoft Word" becomes the title.
Type The symbol and buttons displayed in the box. It is the sum of values from the following groups.

Type	Value	Meaning
Button	0	OK button (default).
	1	OK and Cancel buttons.
	2	Abort, Retry, and Ignore buttons.
	3	Yes, No, and Cancel buttons.
	4	Yes and No buttons.
	5	Retry and Cancel buttons.
Icons	0	No icon (default).
	16	Stop icon.
	32	Question icon.
	48	Attention icon.
	64	Information icon.
Button action	0	First button is the default.
	256	Second button is the default.
	512	Third button is the default.

Examples
MsgBox "Unable to find file", "Microsoft Word", 16
Message box with title of "Microsoft Word", message of "Unable to find file", OK button, and Stop icon (0 + 16 + 0 = 16).
MsgBox "Delete File?", "Librarian", 289
Message box with title of "Librarian", message of "Delete File?", OK button, Cancel button, Question icon, and second (Cancel) button as the default (1 + 32 + 256 = 289).
If Type is a negative value, the message is displayed in the status bar and Type must be -1 (display the message until another message replaces it), -2 (display until a mouse or key action occurs), or -8 (use the entire status bar width until a mouse or key action occurs).
Because MsgBox does not return a value, the use of button values other than 0 (zero) is not recommended. To make use of buttons other than the OK button, use the MsgBox() function.

_____
See also
Dialog Box Statements and Functions

MsgBox()

n = MsgBox(Message$ [,Title$] [,Type])
Creates a dialog box, as in MsgBox, but returns a number describing the button clicked. Message$, Title$, and Type are used the same way they are used in MsgBox.

Returns one of the following values:

Return value	Button pressed	Button text
-1	Leftmost button	OK
		Yes
		Abort
0	Next button	Cancel
		No
		Retry
1	Next button	Cancel
		Ignore

If Type is a negative value, MsgBox() always returns 0 (zero).

_____
See also
Dialog Box Statements and Functions
MsgBox

Name...As

Name OldName$ As NewName$
Renames a file.

OldName$ The previous name of the file, as text.
**NewName$** The new name of the file, as text.. If the file name specified by NewName$ already exists, an error is generated.

Example
Name "COGS.DOC" As "COGS88.DOC"
Renames the file COGS.DOC as COGS88.DOC.

_____
See also
File Statements and Functions

NextCell

NextCell
logical = NextCell()
Selects the next cell in a table. If more than one cell is already selected, selects contents of first cell in selection. If there is no next cell, a new row is added to the table.
The function form returns 0 (zero) if there is no next cell. Determining this is useful for avoiding the addition of a new row in a table.

_____
See also
Selection Statements and Functions
Table Statements and Functions

NextField

NextField
logical = NextField()
Moves the selection to the next field that has a result. Skips over marker fields, such as Index Entry {XE}, Table of Contents {TC}, and Referenced Doc. (RD) fields.
The function form returns 0 (zero) if there is no next field.

_____
See also
Field Statements and Functions
Selection Statements and Functions

NextObject

NextObject
Selects the next text area in page layout view. You can identify text areas by selecting Text Boundaries, a View option in the Options dialog box (Tools menu).

_____
See also
Selection Statements and Functions

NextPage

NextPage
logical = NextPage()
Displays the beginning of the next page in page layout view.
The function form returns 0 (zero) if the action cannot be completed.

_____
See also
Selection Statements and Functions
View Statements and Functions

NextTab()

n = NextTab(Pos)
Returns the position of the next tab stop to the right of the position given by Pos, in points, for the first paragraph in the selection.

Example
firstTabType = TabType(NextTab(0))
Returns the type of the first tab in the selection.

_____
See also
Formatting Statements and Functions

NextWindow

NextWindow
Activates the next window.

_____
See also
Window Statements and Functions

NormalStyle

NormalStyle
n = NormalStyle()
Formats the selected paragraphs with the Normal style.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

OK

OK
Terminates a CopyFormat or CopyMove operation and performs its action.

_____

See also
Editing Statements and Functions
Cancel

OKButton

OKButton x, y, dx, dy
Used as part of a user dialog definition, creates an OK button that the user chooses to terminate the dialog box. If the user chooses the OK button, the macro continues.

x, y	The coordinates of the upper-left corner of the OK button, in units of 1/8th (for x) and 1/12th (for y) of the system font, relative to the upper-left corner of the dialog box.
dx, dy	The width and height of the OK button, in units of 1/8th (for dx) and 1/12th (for dy) of the system font. For more information, see "Using Macros" in the Microsoft Word User's Guide.

_____

See also
Dialog Box Statements and Functions
CancelButton
Err
Error
On Error
PushButton

On Error

On Error Goto Label
On Error Resume Next
On Error Goto 0
The On Error control structure sets an error "trap" that causes a specific action to execute when an error occurs. The influence of an On Error instruction is local to the portion of the program in which it occurs. It is possible (and necessary, if you want to trap all errors) for the main line and each subroutine of a program to have their own On Error instruction and error-handling routine. When a subroutine is called, the local On Error instruction (if any) is in effect. After control is returned to the main program, the main On Error instruction is again in effect. It is not necessary to repeat the main On Error instruction after calling a subroutine.
The form On Error Goto Label jumps from the line where the error occurred to the specified label.
The instructions following this label can then determine the nature of the error (using the special Err variable) and take some appropriate action to correct or resolve the problem (see Err).
The form On Error Resume Next is the simplest type of error handling possible. If an error occurs, the program continues from the line that follows the line where the error occurred, and sets Err to 0. In effect, the error is ignored. This statement should be used with caution because no error message is displayed and the condition that caused the error may still exist.
On Error Goto 0 disables the error trapping established by an earlier On Error Goto or On Error Resume Next instruction, and sets Err to 0.
Once an error trap is sprung by an error, no further error trapping occurs until the trap is reset. This is done using an Err = 0 instruction. An Err = 0 instruction usually should be placed at the exit points of your error-handling routine. Such placement means that if an error occurs in the error routine, a message will be displayed and the macro will halt; this prevents the possibility of an endless loop. Any OnError instruction will reset Err to 0. Such a loop would happen in a macro where Err = 0 is located at the beginning of the error routine and is followed by an error. The error causes a jump to the beginning of the error-handling routine where the error trap is reset and the error occurs again. This cycle could go on indefinitely, until the macro is interrupted by the user.
Errors with numbers 1,000 or greater are generated by Word and not the macro language itself. If such an error occurs, an error

message box is displayed, and the user must respond before the macro can continue. If the user chooses OK, control of the program then passes to the error-handling statements.

_____

See also

Standard Basic Statements and Functions

OnTime

OnTime When$, Name$ [,Tolerance]
Sets up a background timer that runs a specified macro when the time has elapsed.

When$	The time the macro is to be run, expressed as text in a 24-hour format, optionally preceded by a string representing the date. If the date is not specified, the macro runs at the first occurrence of the specified time.
Name$	The name of the macro to be run.
Tolerance	Word does not run the macro if more than Tolerance seconds have elapsed since When$, and the macro has not yet run. If Tolerance is 0 (zero) or omitted, Word always runs the macro, regardless of how many seconds elapse before Word is idle and available to run the macro. The macro is executed the next time Word is idle after the specified When$. Word can run only one OnTime macro at a time. If you start a second, it replaces the first as the background macro, and the first stops running without completing.

Examples
The following example sets up a simple alarm clock function in Word. The first macro sets up the background timer:

```
        'Alarm program: Prompts user to input time for alarm to sound.
        'Current time appears in title bar of input box.
Sub MAIN
  Alarm$ = InputBox$("Time? (HH:MM:SS), 24hr", "Alarm " + Time$())
        'Set background timer to run macro called Beeper.
        'No tolerance is set, so alarm will always sound.
  OnTime Alarm$, "Beeper"
End Sub
```

The following macro sounds an alarm and an alert message in response to the timer set up by the preceding macro. This macro must be named Beeper to work properly.

```
Sub MAIN                                'Beeper program
  For Count = 1 To 7                    'Seven series of beeps
    Beep                                        'Beep once
    Beep                                        'Beep twice
    For TL = 1 To 100                   'Timer loop to wait a while
    Next                                        'Next timer loop
  Next                                          'Next Count
                                                'Display msg box with current time in title bar
  MsgBox "Preset alarm sounded", "Beeper " + Time$(), 48
End Sub
```

_____
See also
Standard Basic Statements and Functions

OpenUpPara

OpenUpPara
Changes the Space Before option in the Paragraph dialog box (Format menu) to one line for the current paragraph(s).

_____
See also
Formatting Statements and Functions

Open...For...As

Open Name$ For Mode$ As [#]StreamNumber
Opens the specified file or device, for input or output of text.

Name$	The name of the file to open or a device, such as Com1 or Lpt1. Do not include the colon following the device name.

Mode$	**The mode in which the file is opened:**
Input	Opens the file for input.
Output	Opens the file for output, replacing previous data in the file.
Append	Opens the file for appending output, or creates a new file.

StreamNumber	A file stream number; an integer from 1 through 4.

_____
See also
File I/O Statements and Functions

OptionButton

OptionButton x, y, dx, dy, Text$
Defines an option button within a dialog box.

x, y	The coordinates of the upper-left corner of the rectangle containing both the option button and its associated text, in units of 1/8th (for x) and 1/12th (for y) of the system font.
dx, dy	The width and height of the rectangle, in units of 1/8th (for dx) and 1/12th (for dy) of the system font.
Text$	The text label associated with the option button. An ampersand (&) preceding a character in Text$ makes that character the underlined access key for selecting the option button. Within a group of option buttons, only one button may be active at a time. An OptionGroup instruction begins the definition of a series of related option buttons. For more information, see "Using Macros" in the Microsoft Word User's Guide.

Example
...other instructions
OptionGroup .brk
OptionButton 63, 36, 24, 12, "&PageBreak"
OptionButton 63, 47, 24, 12, "&LineBreak"
...other instructions

_____
See also
Dialog Box Statements and Functions

OptionGroup

OptionGroup .Field
Begins the definition of a series of related option buttons. Within the group, only one button can be active at a time.

.Field	Defines a variable that represents the number of the active button within the option group. 0 (zero) corresponds to the first button, 1 to the second button, and so on. For more information, see "Using Macros" in the Microsoft Word User's Guide.

Example
...other instructions
OptionGroup .brk
OptionButton 63, 36, 24, 12, "&PageBreak"
OptionButton 63, 47, 24, 12, "&LineBreak"
...other instructions

_____
See also
Dialog Box Statements and Functions

OtherPane

OtherPane
Activates the other pane of the active window.

_____
See also
Window Statements and Functions

OutlineCollapse

OutlineCollapse
Collapses one level of text under the selected headings.

_____
See also
Outlining Statements and Functions

OutlineDemote

OutlineDemote
Applies the next higher-numbered heading level style to the selected paragraphs.

_____
See also

OutlineExpand

OutlineExpand
Expands one level of text under the selected headings.

_____
See also
Outlining Statements and Functions

OutlineLevel()

n = OutlineLevel()
Returns the heading level of the selected paragraph. Returns 0 (zero) if the selected paragraph has no defined level (for example, if it is body text). If multiple paragraphs are selected, returns the level of the first paragraph in the selection.

_____
See also
Outlining Statements and Functions

OutlineMoveDown

OutlineMoveDown
Moves the selection below the next visible paragraph.

_____
See also
Outlining Statements and Functions

OutlineMoveUp

OutlineMoveUp
Moves the selection above the previous visible paragraph.

_____
See also
Outlining Statements and Functions

OutlinePromote

OutlinePromote
Applies the next lower-numbered heading level style to the selected paragraphs.

_____
See also
Outlining Statements and Functions

OutlineShowFirstLine

OutlineShowFirstLine [On]
Changes the view of nonheading text between all text shown and only the first line of text shown.

On	Specifies whether or not to display the first line of text:
0	All nonheading text is shown.
1	Only the first line of text is shown.

_____
See also
Outlining Statements and Functions

Overtype

Overtype [On]
logical = Overtype()
Switches between overtype and insert modes.

On	Specifies the mode:
0	Overtype mode is off (insert mode is on).
1	Overtype mode is activated and the letters OVR are displayed in the status bar.
omitted	Toggles overtyping mode.

The function form returns:

0 (zero)	If overtype mode is off.
-1	If overtype mode is on.

_____

See also
Environment Statements and Functions

PageDown

PageDown [Count,] [Select]
logical = PageDown([Count,] [Select])
Corresponds to the PAGE DOWN key; moves the insertion point or selection down by the specified number of screens (equal to the height of the active window).

Count	The number of screens to scroll; if omitted, 1 is assumed.
Select	If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended.
	The function form returns -1 if operation was successful or 0 (zero) if not.

Example
PageDown 1, 1
Extends the selection down one screen from the insertion point.

_____
See also
Selection Statements and Functions

PageUp

PageUp [Count,] [Select]
logical = PageUp([Count,] [Select])
Corresponds to the PAGE UP key; moves the insertion point or selection up by the specified number of screens (equal to the height of the active window).

Count	The number of screens to scroll; if omitted, 1 is assumed.
Select	If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended.
	The function form returns -1 if operation was successful or 0 (zero) if not.

Example
PageUp 20
Moves the insertion point up 20 screens, but does not extend the selection.

_____
See also
Selection Statements and Functions

ParaDown

ParaDown [Count,] [Select]
logical = ParaDown([Count,] [Select])
Moves the insertion point or extends the selection down by the specified number of paragraphs.

Count	The number of paragraphs to move; if omitted, 1 is assumed.
Select	If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended.
	The function form returns 0 (zero) if the action cannot be performed; for example, if the insertion point is at the end of the document.

Example
While ParaDown(,1) : Wend
Extends the selection one paragraph at a time until the end of the document is reached.

_____
See also
Selection Statements and Functions

ParaUp

ParaUp [Count,] [Select]
logical = ParaUp([Count,] [Select])
Moves the insertion point or extends the selection up by the specified number of paragraphs.

Count	The number of paragraphs to move; if omitted, 1 is assumed.
Select	If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended. The function form returns 0 (zero) if the action cannot be performed; for example, if the insertion point is at the beginning of the document.

_____
See also
Selection Statements and Functions

PauseRecorder

PauseRecorder
Stops macro recording until PauseRecorder is executed again.

_____
See also
Macro Statements and Functions

PrevCell

PrevCell
logical = PrevCell()
Moves the selection to the previous cell. If the selection is more than one cell, selects the first cell in the selection.
The function form returns 0 (zero) when the selection is in the first cell, and nonzero otherwise.

_____
See also
Selection Statements and Functions
Table Statements and Functions

PrevField

PrevField
logical = PrevField()
Moves the selection to the previous field.
The function form returns 0 (zero) when the selection is in the first field, and nonzero otherwise.

_____
See also
Field Statements and Functions
Selection Statements and Functions

PrevObject

PrevObject
Moves to the previous text area in page layout view. You can identify text areas by selecting Text Boundaries, a View option in the Options dialog box (Tools menu).

_____
See also
Selection Statements and Functions

PrevPage

PrevPage
logical = PrevPage()
Displays the beginning of the previous page in page layout view.
The function form returns 0 (zero) if the action cannot be completed.

_____
See also
Selection Statements and Functions
View Statements and Functions

PrevTab()

n = PrevTab(Pos)
Returns the position of the previous tab stop to the left of the position given by Pos, in points, for the first paragraph in the selection.
If more than one paragraph is selected and the previous tab positions do not all match, returns -1.

Example
lastTabType = TabType(PrevTab(72*22))
Returns the type of the last tab in the selection (at 72 points per inch times 22 inches).

_____
See also
Formatting Statements and Functions

PrevWindow

PrevWindow
Activates the previous window.

_____
See also
Window Statements and Functions

Print

Print [[#]StreamNumber,] Expression
Writes Expression to the file specified by StreamNumber. With no StreamNumber specified, output goes to the status bar.

Examples
Print #1, ItemA$, ItemB$
Writes contents of ItemA$ and ItemB$ to file opened as #1. Items are inserted into a single paragraph and are separated by a tab.
Print "Total is "; TotalSales
Displays "Total is" followed by the value of TotalSales in the status bar.

_____
See also
File I/O Statements and Functions

PushButton

PushButton x, y, dx, dy, Text$
Creates a push button within a dialog box.

x, y	The coordinates of the upper-left corner of the rectangle containing the pushbutton, in units of 1/8th (for x) and 1/12th (for y) of the system font, relative to the upper-left corner of the dialog box.
dx, dy	The width and height of the push button, in units of 1/8th (for dx) and 1/12th (for dy) of the system font.
Text$	The associated text label for the push button. An ampersand (&) preceding a character in Text$ makes that character the underlined access key for choosing the push button.

Example
PushButton 40, 20, 80, 18, "&Create"
For more information, see "Using Macros" in the Microsoft Word User's Guide.

_____
See also
Dialog Box Statements and Functions
Begin Dialog...End Dialog

Read

Read [#]StreamNumber, Variable(s)
Similar to the Input statement as used for file access, but removes quotation marks from strings. This statement is used with the Write statement.

_____
See also
File I/O Statements and Functions

RecordNextCommand

RecordNextCommand
Records the next command and inserts it at the insertion point in the current macro window.

_____
See also
Macro Statements and Functions

Redim

Redim [Shared] Var [(Size)] [, Var [(Size)]]
Reallocates storage space for a previously defined variable array. May be used to enlarge an array, but array contents will be lost. Redim can also be employed to reuse a previously defined dialog record.

_____
See also
Standard Basic Statements and Functions
Dim

Rem

Rem Remarks
'Remarks
Inserts explanatory text into the macro. You can use an apostrophe (') instead of Rem. If Rem follows another instruction on a line, it must be separated from the instruction by a colon (:). A colon is not required before a remark introduced by an apostrophe.

_____
See also
Standard Basic Statements and Functions

RemoveFrames

RemoveFrames
Removes all frames in the selection.

_____
See also
Editing Statements and Functions

RenameMenu

RenameMenu MenuNumber, NewText$
Renames the specified menu until the user quits Word.

MenuNumber	**The number of the menu to renumber:**
0	File
1	Edit
2	View
3	Insert
4	Format
5	Tools
6	Macro
7	Window

NewText$	The new menu name. An ampersand (&) preceding a character in the menu name sets an underlined access key for choosing the menu command.

Examples
RenameMenu 5, "&Other Tasks",
Changes Tools to Other Tasks, with O being the key that activates the menu.

_____
See also
Tools Statements and Functions
View Statements and Functions

RepeatFind

RepeatFind
Repeats Go To or Find to find the next occurrence. Repeats the most recent find operation.
Corresponds to pressing SHIFT+F4.

_____
See also
Editing Statements and Functions

ResetChar

ResetChar
n = ResetChar()
Removes manual character formatting from the selected text. Manual character formatting is formatting that is not applied as a style. For example, you manually format a word or phrase in a paragraph as bold text and the paragraph style is normal text. ResetChar would return the bold text to the character format dictated by the style.

The function form returns the following values without resetting character formats:

0 (zero)	If any manual character formatting is present in the selection.
1	If the selected text contains no manual character formatting.

_____
See also
Formatting Statements and Functions

ResetFootnoteContNotice

ResetFootnoteContNotice
Resets the footnote continuation notice to the default value. Works only in the footnote continuation notice pane (see InsertFootnote) and only if the contents of the pane are "dirty" (that is, changes have been made since the last save).

_____
See also
Formatting Statements and Functions

ResetFootnoteContSep

ResetFootnoteContSep
Resets the footnote continuation separator to the default value. Works only in the footnote continuation separator pane (see InsertFootnote) and only if the contents of the pane are "dirty" (that is, changes have been made since the last save).

_____
See also
Formatting Statements and Functions

ResetFootnoteSep

ResetFootnoteSep
Resets the footnote separator to the default value. Works only in the footnote separator pane (see InsertFootnote) and only if the contents of the pane are "dirty" (that is, changes have been made since the last save).

_____
See also
Formatting Statements and Functions

ResetPara

ResetPara
n = ResetPara()
Removes manual paragraph formatting from the selected text. The text is left with the paragraph formatting of the current style.

The function form returns the following values, without removing manual paragraph formats:

0 (zero)	If any manual paragraph formatting is present.
-1	If the selected text contains no manual paragraph formatting.

_____
See also
Formatting Statements and Functions

Right$()

a$ = Right$(Source$, Count)
Returns the rightmost Count characters of Source$.

Examples

```
a$ = "Legal File List"
Print Right$(A$,4)
```

Prints List in the status bar.

```
a$ = "Legal File List"
Print Right$(A$,9)
```

Prints File List in the status bar.

_____
See also
Standard Basic Statements and Functions

RightPara

RightPara
n = RightPara()
Right-aligns the selected paragraphs.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of alignments.
1	If all of the selection is in the format.

_____
See also
Formatting Statements and Functions

RmDir

RmDir Name$
Removes the specified directory or subdirectory. Files must first be removed from the subdirectory for this statement to work. RmDir cannot remove the current subdirectory.

Example

```
Kill "C:\WORD\FCCPROJ\*.*"      'Delete all files in directory.
RmDir "C:\WORD\FCCPROJ"         'Remove empty directory.
```

_____
See also
File Statements and Functions

Rnd()

n = Rnd([Expression])
Returns a random fractional value between 0 (zero) and 1. The Expression is not used by WordBasic but is provided for compatibility with other forms of BASIC.

Examples
a = Rnd()
A random value between 0 (zero) and 1 is stored in a.
a = Int(Rnd() * 10)
A random integer between 0 (zero) and 10 is stored in a.

_____
See also
Standard Basic Statements and Functions

RulerMode

RulerMode
Activates and deactivates the ruler for use with keys, permitting the setting of indents and tabs from the keyboard. Equivalent to pressing CTRL+SHIFT+F10.

_____
See also
Environment Statements and Functions

SaveTemplate

SaveTemplate
Saves the document's template. If the document has no template, saves NORMAL.DOT

_____
See also
File Statements and Functions

Seek

Seek [#]StreamNumber, Count
n = Seek([#]StreamNumber)
Positions a file pointer at character Count in the file attached to the stream specified by StreamNumber.
The function form returns the current file pointer for the specified StreamNumber.

_____
See also
File I/O Statements and Functions

Select Case...Case Else...End Select

Select Case Expression
Case CaseExpression
Statement(s)
[Case Else
Statement(s)]
End Select
The Expression is compared with all the values given in each CaseExpression until a match is found. If a match is found, the instruction or instructions following each CaseExpression is executed. If there is no match and there is a Case Else instruction, the instructions following it are executed. If there is no match and there is no Case Else instruction, an error occurs.

Example

```
Select Case Int(Rnd() * 10) - 5
Case 1,3
  Print "One or three"
Case Is > 3
  Print "Greater than three"
Case -5 to 0
  Print "Between -5 and 0 (inclusive)"
Case Else
  Print "Must be 2"
End Select
```

The Case expression generates a number between -5 and 5, and the subsequent Case instructions are executed depending on the value of the expression.

_____
See also
Standard Basic Statements and Functions

SelectCurAlignment

Starting at the insertion point and moving toward the end of the document, selects adjacent paragraphs having the same alignment as the selected text.

_____
See also
Selection Statements and Functions

SelectCurColor

Starting at the insertion point and moving toward the end of the document, selects adjacent characters having the same color as the selected text.

_____
See also
Selection Statements and Functions

SelectCurFont

Starting at the insertion point and moving toward the end of the document, selects adjacent characters having the same font as the selected text.

_____

See also

Selection Statements and Functions

SelectCurIndent

Starting at the insertion point and moving toward the end of the document, selects adjacent paragraphs having the same indention as the selected text.

_____

See also

Selection Statements and Functions

SelectCurSpacing

Starting at the insertion point and moving toward the end of the document, selects adjacent paragraphs having the same line spacing as the selected text.

_____

See also

Selection Statements and Functions

SelectCurTabs

Starting at the insertion point and moving toward the end of the document, selects adjacent paragraphs having the same tab stops as the selected text.

_____
See also
Selection Statements and Functions

Selection$()

a$ = Selection$()
Returns the plain, unformatted text of the selection. The selection can contain up to 32,000 characters, or the maximum that memory can hold. If the selection is too large, Selection$() is filled with as much of the selection as possible, and an error is generated. If the selection is an insertion point, the character following the insertion point is returned.

Example

```
Sub MAIN
  On Error Goto TooBig                    'Set up error trap
  EditSelectAll                           'Select entire document
  all$ = Selection$()                'Store text of selection as All$
  EditCut                            'Delete selected text
  upper$ = UCase$(all$)                   'Convert text to all uppercase
  Insert upper$                           'Insert text back into document
  Goto Finish

TooBig:
  If Err = 513 Then                       'If error is "String too long"
    MsgBox "Sorry, document too big"      '"Too big" error message
  Else                                    'Otherwise...
    Error Err                             'Unknown: display error msg
  End If

Finish:                                   'End label
End Sub
```

_____
See also
Selection Statements and Functions

SelInfo()

n = SelInfo(Type)
Returns various types of information about the selection.

Type	The type of information to return:
1	Number of the page containing the selection. If the selection is in a header or footer pane in normal view, returns -1. If the selection is in a footnote or annotation pane, returns the page number of the previous footnote or annotation.
2	Number of the section containing the selection.
3	Number of the page in the document containing the selection, where the first page is number 1, and so on.
4	Number of pages in the selection.
	Type 5 is valid only in page layout view. Type 6 is valid in page layout view, or in normal view if Line Breaks And Fonts As Printed and Background Repagination are selected in the Options dialog box (Tools menu).
5	Horizontal position of the selection; distance between the left edge of the selection and the left edge of the page, in twips (1/20th of a point, or 1/1,440th of an inch). Returns -1 if the selection is not visible.
6	Vertical position of the selection; distance between the top edge of the selection and the top edge of the page, in twips. Returns -1 if the selection is not visible.
7	Horizontal position of the selection, relative to the left edge of the display boundary enclosing it, in twips. Returns -1 if the selection is not visible.
8	Vertical position of the selection, relative to the upper edge of the display boundary enclosing it, in twips. Returns -1 if the selection is not visible.
9	Number of characters between the first character in the selection and the beginning of the current line (same as the character column number displayed in the status bar).
10	The line number of the first character in the selection; if Background Repagination and Line Breaks And Fonts As Printed are not in effect, returns -1.
11	Returns -1 if the selection is an entire frame.
12	Returns -1 if the selection is in a table.
	Types 13 through 18 apply only if the selection is within a table; if the selection is not in a table, the function returns -1.
13	The row containing the beginning of the selection.
14	The row containing the end of the selection.
15	The number of rows in the table.
16	The column containing the beginning of the selection.
17	The column containing the end of the selection.
18	The maximum number of columns within any row in the selection.
19	The current zoom factor.
20	The type of selection: returns 0 for normal selection, 1 for extended selection, and 2 for block selection. Corresponds to the box in the status bar that reads either EXT or COL.
21	Returns -1 if Caps Lock is in effect.
22	Returns -1 if Num Lock is in effect.
23	Returns -1 if Word is in overtype mode.
24	Returns -1 if revision marking is in effect.
25	Returns -1 if the selection is in the footnote pane or in a footnote in page layout view.
26	Returns -1 if the selection is in an annotation pane.
27	Returns -1 if the selection is in a macro editing window.
28	Returns -1 if the selection is in the header/footer pane or in a header or footer in page layout view.
29	The number of the bookmark enclosing the start of the selection; 0 if none or invalid.
30	The number of the last bookmark that starts before selection (not necessarily including the selection); returns 0 if none or invalid.

_____

See also
Selection Statements and Functions

SelType

SelType Type
n = SelType()
Changes the selection highlighting to the specified form.

Type	The type of highlighting:
0	Hidden
1	Solid insertion point (default)
2	Solid selection (default)
4	Dotted selection or insertion point (whichever is current)
5	Dotted insertion point
6	Dotted selection

The function form returns the type of the selection highlighting, according to the table above.

_____

See also

Selection Statements and Functions

SendKeys

SendKeys Keys$, Wait
Sends the keys specified by Keys$ to the active application, just as if they were typed at the keyboard.
SendKeys must be executed before the program or command that receives the keystrokes so that those keystrokes are in the buffer when the macro pauses.

Keys$	A key sequence, such as a for the letter a, {enter} for the ENTER key, and {pgup} for the PAGE UP key.
Wait	If Word is not the active application and Wait is -1, Word waits for all keys to be processed before proceeding.
	To specify characters that aren't displayed when you press the key, use the codes shown in the following table.

Key	**Code**
BACKSPACE	{backspace} or {bs} or {bksp}
BREAK	{break}
CAPS LOCK	{capslock}
CLEAR	{clear}
DEL	{delete} or {del}
DOWN ARROW	{down}
END	{end}
ENTER	{enter}
ESC	{escape} or {esc}
HELP	{help}
HOME	{home}
INS	{insert}
LEFT ARROW	{left}
NUM LOCK	{numlock}
PAGE DOWN	{pgdn}
PAGE UP	{pgup}
PRINT SCREEN	{prtsc}
RIGHT ARROW	{right}
TAB	{tab}
UP ARROW	{up}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
F13	{F13}
F14	{F14}
F15	{F15}
F16	{F16}

The plus sign (+), the percent sign (%), and the caret (^) have special meanings.

To combine with	**Precede the keycode by**
SHIFT	+ (plus sign)
ALT	% (percent sign)
CTRL	^ (caret)

Note:	When sending key combinations that include the ALT key, make it a rule to send lowercase characters. For example, to open the File menu (ALT, F), send %f -- %F sends ALT+SHIFT+F.To repeat a key sequence, use the syntax {Keys$ Number}; remember to put a space between the key and the number.

Examples

For example, %{enter} sends the code for ALT+ENTER. You can group keys with parentheses and precede the group with the keycode for a SHIFT, ALT, or CTRL key. For example, the code +(eb) specifies EB (but you can also simply use the uppercase letters EB, without using +).

SendKeys "s"
Help
Opens or activates Help and then displays the Search dialog box.
Key$ = "{break}"
SendKeys Key$, -1
AppActivate "Microsoft Excel"
Pauses an operation in Microsoft Excel.
SendKeys "{pgdn 20}"
Shell "MARCH.XLS", 1
Starts Microsoft Excel with the worksheet MARCH.XLS and then performs the equivalent to pressing the PAGE DOWN key 20 times.

Note Use great care when applying SendKeys to operate other programs. Word has no way to detect or correct errors generated by the other program and always sends the programmed series of keystrokes. Be sure to test your macros with the other program under a variety of conditions to ensure that the keystrokes required by the other program remain exactly the same. If the other program requires different keystrokes than those programmed, data could be lost. If more keystrokes are required than are programmed (as in an {enter} to respond to an error message box), the macro freezes, waiting for the missing input. If there are too many keystrokes in the programmed sequence, any extra are passed to Word itself, and the results are unpredictable. Use SendKeys to operate other applications only when there is no alternative, and then use it with caution. In general, DDE (Dynamic Data Exchange) is a better way for Word to interact with other programs, because DDE provides a channel for two-way communication between programs and provides a path for detecting and dealing with errors in the other application.

_____

See also
Dynamic Data Exchange (DDE) Statements and Functions

SentLeft

SentLeft [Count,] [Select]
logical = SentLeft([Count,] [Select])
Moves the insertion point or extends the selection left by the specified number of sentences.

Count The number of sentences to move; if omitted, 1 is assumed.
Select If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended.

The function form returns 0 (zero) if the action cannot be performed; for example, if the insertion point is at the beginning of the document.

_____

See also
Selection Statements and Functions

SentRight

SentRight [Repeat,] [Select]
logical = SentRight([Repeat,] [Select])
Moves the insertion point or extends the selection right by the specified number of sentences.

Count The number of sentences to move; if omitted, 1 is assumed.
Select If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended.

The function form returns 0 (zero) if the action cannot be performed; for example, if the insertion point is at the end of the document.

_____

See also
Selection Statements and Functions

SetDirty

SetDirty [Dirty]
Causes Word to recognize the active document as "dirty" (that is, changed since the last time the document was saved) or to remove this attribution, in order to avoid a dialog box requesting whether or not to save the changes made in the document.

Dirty	**Specifies whether or not to make the document "dirty":**
0	The document is treated as unchanged (that is, "clean")

1 or omitted	The document is recognized as dirty. The dirty flag is also used by Word functions such as FileClose and FileSaveAll.

_____

See also
Environment Statements and Functions
File Statements and Functions
IsDirty()

SetEndOfBookmark

SetEndOfBookmark Bookmark1$ [,Bookmark2$]
Sets Bookmark2$ to the end point of Bookmark1$. If Bookmark2$ is omitted, Bookmark1$ is set to its own end.

_____
See also
Bookmark Statements and Functions

SetGlossary

SetGlossary Name$, Text$ [,Context]
Defines a glossary entry.

Name$	The name of the new entry.
Text$	The text to be associated with the entry.

Context	**Scope of application:**
0 (zero)	Global (default)
1	Document template

Example
text$ = "No warranty is either expressed or implied."
SetGlossary "Disclaim", text$, 1
Defines the glossary entry Disclaim in the template context, containing the text specified in text$.

_____
See also
Glossary Statements and Functions

SetProfileString

SetProfileString [App$,] Key$, Value$
Defines or redefines a value in the current WIN.INI file.

App$	The name of the Microsoft Windows application. If the application is not specified, the string "Microsoft Word 2.0" is used.
Key$	The key to define.
Value$	The value assigned to the key.

Example
SetProfileString "programdir", "C:\WINWORD2"
Assigns the value C:\WINWORD2 to the key programdir.

_____
See also
Environment Statements and Functions

SetStartOfBookmark

SetStartOfBookmark Bookmark1$ [,Bookmark2$]
Sets Bookmark2$ to the starting point of Bookmark1$. If Bookmark2$ is omitted, Bookmark1$ is set to its own starting point.

_____
See also
Bookmark Statements and Functions

Sgn()

n = Sgn(n)

The function returns:

0 (zero)	If n is zero.
-1	If n is a negative number.
1	If n is a positive number.

_____
See also
Standard Basic Statements and Functions

Shell

Shell App$ [,WindowStyle]
Starts another program under Microsoft Windows.

App$	The exact name of the program file and the extension and path required to find the file, as well as any switches or arguments that the program accepts. If App$ is the name of a file with an extension specific to an installed application that is specified in the WIN.INI file (for example, .DOC for a Word document), Shell starts the application and loads the file.

WindowStyle	**How the window containing the program should appear (some programs ignore this):**
0	Minimized window
1	Normal window
2	Minimized window (for Microsoft Excel compatibility)
3	Maximized window
4	Deactivated window

Examples

Shell "EXCEL.EXE", 2
Starts Microsoft Excel and minimizes the window.
Shell "TRENDS.XLC", 1
Starts Microsoft Excel and loads the document TRENDS.XLC in a normal window.
Shell "NOTEPAD.EXE TORT.TXT"
Starts Notepad and loads the document TORT.TXT.

_____
See also
Environment Statements and Functions
File Statements and Functions

ShowAll

ShowAll [On]
logical = ShowAll()
Corresponds to selecting the All check box in the View category in the Options dialog box (Tools menu); displays all nonprinting characters, such as hidden text, tabs, spaces, and paragraph marks.

On	**Specifies whether to hide or display all nonprinting characters:**
0	Hides nonprinting characters.
1	Displays nonprinting characters.
omitted	Toggles the option.

The function form returns:

0	If nonprinting characters are not displayed.
-1	If nonprinting characters are visible.

_____
See also
View Statements and Functions
ToolsOptionsView

ShowAllHeadings
ShowAllHeadings
Shows all text in outline view.

_____
See also
Outlining Statements and Functions

ShowHeading1

ShowHeading1
In outline view, shows all level 1 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading2

ShowHeading2
In outline view, shows all level 1 and 2 headings and hides subordinate headings and body text.

_____
See also

ShowHeading3

ShowHeading3
In outline view, shows all level 1 through 3 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading4

ShowHeading4
In outline view, shows all level 1 through 4 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading5

ShowHeading5
In outline view, shows all level 1 through 5 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading6

ShowHeading6
In outline view, shows all level 1 through 6 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading7

ShowHeading7
In outline view, shows all level 1 through 7 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading8

ShowHeading8
In outline view, shows all level 1 through 8 headings and hides subordinate headings and body text.

_____
See also
Outlining Statements and Functions

ShowHeading9

ShowHeading9
In outline view, shows all level 1 through 9 headings and hides body text.

_____
See also
Outlining Statements and Functions

ShowVars

ShowVars
Corresponds to clicking the Vars button in the macro editing bar; displays a list of variables (and their values) currently in use. This statement is useful for debugging macros.

_____
See also
Macro Statements and Functions

ShrinkFont

ShrinkFont
Decreases the size of the selected font to the next available size supported by the assigned printer.
Can be used either for the selection or at the insertion point. If the selection contains more than one font size, each is reduced to its next available size.

_____
See also
Formatting Statements and Functions

ShrinkSelection

ShrinkSelection

Shrinks the selection to the next smaller unit (word, sentence, paragraph, and so forth).

_____

See also
Selection Statements and Functions

SmallCaps

SmallCaps [On]
n = SmallCaps()
Adds or removes the Small Caps character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

SpacePara1

SpacePara1
n = SpacePara1()
Formats the selected paragraphs with single spacing (height of the largest font used).

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of other line spacings.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

SpacePara2

SpacePara2
n = SpacePara2()
Formats the selected paragraphs with double spacing (height of the largest font used plus 12 points).

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of other line spacings.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

SpacePara15

SpacePara15
n = SpacePara15()
Formats the selected paragraphs with one-and-one-half line spacing (height of the largest font used plus 6 points).

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of other line spacings.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

Spike

Spike
Copies the selection to a special glossary called the Spike and then deletes the selection.

_____

See also
Editing Statements and Functions
Glossary Statements and Functions

StartOfColumn

StartOfColumn [Select]
logical = StartOfColumn([Select])
Moves the insertion point to the uppermost position in the first selected table column. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the top of the column.
-1	If the action was performed.

_____

See also
Selection Statements and Functions
Table Statements and Functions

StartOfDocument

StartOfDocument [Select]
logical = StartOfDocument([Select])
Moves the insertion point to the beginning of the document. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the start of the document.
-1	If the action was performed.

_____

See also
Selection Statements and Functions

StartOfLine

StartOfLine [Select]
logical = StartOfLine ([Select])
Moves the insertion point to the beginning of the line. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the beginning of the line.
-1	If the action was performed.

_____

See also
Selection Statements and Functions

StartOfRow

StartOfRow [Select]
logical = StartOfRow ([Select])
Moves the insertion point to the leftmost position in the first selected table row. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the beginning of the row.
-1	If the action was performed.

_____

See also
Selection Statements and Functions
Table Statements and Functions

StartOfWindow

StartOfWindow [Select]
logical = StartOfWindow([Select])
Moves the insertion point to the upper-left corner of the window in normal view. If Select is nonzero, the selection is extended.

The function form returns:

0	If the insertion point is already at the upper-left corner of the window.
-1	If the action was performed.

_____

See also
Selection Statements and Functions
Window Statements and Functions

Stop

Stop
Stops a running macro and displays a message that says the macro was interrupted; changes the Start button in the macro editing bar to Continue. Useful for debugging macros in an open editing window.

_____
See also
Standard Basic Statements and Functions

Str$()

a$ = Str$(n)
Returns the string representation of the value n. Positive numbers have a leading space character.

_____

See also
Standard Basic Statements and Functions

Strikeout

Strikeout [On]
n = Strikeout()
Adds or removes the Strikeout character format from the selected text.

On	Specifies whether to add or remove the format:
nonzero	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

String$()

a$ = String$(Count, Source$)
a$ = String$(Count, CharCode)
Returns the first character in Source$, repeated Count times. Replacing Source$ with an ANSI character code returns the corresponding ANSI character repeated Count times.

Examples
Print String$(5, 100)
Results in the text ddddd.
Text 0, 100, 321, 13, String$(60, "_")
In a dialog box definition, creates a line of underscore characters across the dialog box.

_____

See also
Standard Basic Statements and Functions

StyleName$()

a$ = StyleName$([Count,] [Context,] [All])
Returns the name of the style defined for the specified context

Count	The number of the style as listed in the style sheet for the context; ranges from 1 through CountStyles(Context). If Count is 0 (zero), the name of the current style is returned; otherwise, the name is taken from the list in the given context.

Context	**Scope of application:**
0 or omitted	Active document
1	Document template
All	If nonzero, all built-in styles are included at the beginning of the list.

See also
Formatting Statements and Functions

SubScript

SubScript [On]
n = SubScript()
Adds or removes the Subscript character format from the selected text.

On	**Specifies whether to add or remove the format:**
nonzero	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

See also
Formatting Statements and Functions

Sub...End Sub

Sub Name ([ParameterList])
...instruction(s)
End Sub
Defines a subroutine.

Name	The name of the subroutine.
ParameterList	A list of arguments, separated by commas. You can then use these arguments in the routine; each argument and its values are isolated from the other routines in the macro.

Example
The following Main routine calls the GoBeep subroutine, passing through the variable NumBeeps the number of times to beep.

```
Sub Main
  NumBeeps = 3
  GoBeep NumBeeps
End Sub

Sub GoBeep(count)
  For n = 1 to count
    Beep
    For t = 1 to 100 : Next
  Next
End Sub
```

You can also have a routine inside one macro call a routine in another macro and pass the routine a value, by preceding the name of the routine with the name of the macro and a period.
In macro MainBeep...

```
Sub Main
  LibMacros.DoBeeps(3)
End Sub
```

In macro LibMacros...

```
Sub DoBeeps(count)
```

```
  For n = 1 to count
    Beep
    For t = 1 to 100 : Next
  Next
End Sub
```

_____

See also
Standard Basic Statements and Functions

Super

Super Statement
If two macros have the same name, Word runs the template macro before a global macro, and a global macro before a built-in command. Super was originally intended to override this order, running the named macro at the next higher context level, but has no effect in the current release of Microsoft Word.

_____

See also
Macro Statements and Functions

SuperScript

SuperScript [On]
n = SuperScript()
Adds or removes the Superscript character format from the selected text.

On	**Specifies whether to add or remove the format:**
nonzero	Formats the selection.
0 (zero)	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

TabLeader$()

a$ = TabLeader$(Pos)
Returns the leader character of the tab at the position Pos, which is given in points. If more than one paragraph is selected and all the tabs do not match, an empty string is returned. The leader characters returned are blank space ( ), period (.), hyphen (-), and underscore (_).

_____

See also
Formatting Statements and Functions

TableColumnWidth

TableColumnWidth [.ColumnWidth = text,] [.SpaceBetweenCols = text,] [.RulerStyle = text,] [.PrevColumn,] [.NextColumn]
Corresponds to the Column Width dialog box (Table menu); sets the column width and the space between columns for the selected cells.

.ColumnWidth	The width of each column.
.SpaceBetweenCols	The distance between the text in each column.
.RulerStyle	Specifies how Word adjusts the columns after the change:
0	Only selected rows are changed.
1	All cells in the selected columns are changed.
2	If the selection is in a column other than the first or last, overall table width is preserved; equivalent to holding down the SHIFT key while dragging a column marker.
3	Changes the width of the selected column and all columns to the right; equivalent to holding down the CTRL key while dragging a column marker. This argument corresponds to recording table column movements with the ruler or through the Column Width dialog box. If the ruler is used, the borders for all rows are moved regardless of the selection. If the dialog is used when recording macros then the format affects only the selected rows.
.PrevColumn	Selects the previous column.
.NextColumn	Selects the next column.

_____

See also
Formatting Statements and Functions
Table Statements and Functions

TableDeleteCells

TableDeleteCells ShiftCells
Corresponds to the Delete Cells dialog box (Table menu); deletes the selected cells.

ShiftCells	Sets the direction to shift the cells to the right or below the selected cells:
0 or omitted	Shift cells left.
1	Shift cells up.
2	Delete entire row.
3	Delete entire column.

_____

See also
Editing Statements and Functions
Table Statements and Functions

TableDeleteColumn

TableDeleteColumns
Deletes the selected columns from a table.

_____

See also
Editing Statements and Functions
Table Statements and Functions

TableDeleteRow

TableDeleteRow
Deletes the selected rows from a table.

_____

See also
Editing Statements and Functions
Table Statements and Functions

TableGridlines

TableGridlines [On]
n = TableGridlines()
Corresponds to selecting the Table Gridlines check box in the View category in the Options dialog box (Tools menu); displays table gridlines.

On	Specifies whether to hide or display table gridlines:
0	Hides table gridlines.
1	Displays table gridlines.
omitted	Toggles the option.

The function form returns:

0	If table gridlines are not displayed.
-1	If table gridlines are visible.

_____

See also
Table Statements and Functions
View Statements and Functions
ToolsOptionsView

TableInsertCells

TableInsertCells .ShiftCells = number
Corresponds to the Insert Cells dialog box (Table menu); inserts cells above or to the left of the selected range of cells in a table.

.ShiftCells	Sets the direction to shift the cells in the selected range:
0 or omitted	Shift cells right.
1	Shift cells down.
2	Insert entire row.
3	Insert entire column.

Example

TableInsertCells .ShiftCells = 2
Inserts a row above the selected cells.

_____

See also
Table Statements and Functions
Editing Statements and Functions

TableInsertColumn

TableInsertColumn
Corresponds to the Insert Columns command on the Table menu; inserts as many columns into a table as are selected.

_____

See also
Editing Statements and Functions
Table Statements and Functions

TableInsertRow

TableInsertRow
Corresponds to the Insert Rows command on the Table menu; inserts as many rows into a table as are selected.

_____

See also
Editing Statements and Functions
Table Statements and Functions

TableInsertTable

TableInsertTable [.ConvertFrom = value,] [.NumColumns = number,] [.NumRows = number,] [.InitialColWidth = text]
Corresponds to the Insert Table dialog box (Table menu); converts a series of selected paragraphs into a table or inserts an empty table if the selection is an insertion point.

.ConvertFrom	Specifies the character used to separate items of text into cell contents:
0	Paragraph marks

1	Tabs
2	Commas

.NumColumns	Number of columns in the table.
.NumRows	Number of rows in the table.
.InitialColWidth	The initial column width, in points, or a measurement in the form of text, or "Auto" for automatic calculation of column width.

Example

```
TableInsertTable .ConvertFrom = 0, .NumColumns = 3, .NumRows = 5,\ .InitialColWidth = "Auto"
```

_____

See also

Formatting Statements and Functions

Table Statements and Functions

TableMergeCells

TableMergeCells
Merges selected table cells in the same row into a single cell. This statement is unavailable if more than one row is selected.

_____
See also
Editing Statements and Functions
Table Statements and Functions

TableRowHeight

TableRowHeight .RulerStyle = text, .LeftIndent = text, .LineSpacingRule = number, .LineSpacing = text, .Alignment = number [,.PrevRow] [,.NextRow]
Corresponds to the Row Height dialog box (Table menu); sets formats for the selected rows in a table.

.RulerStyle	**Specifies whether the entire table or only the selected rows are formatted:**
0	Selected rows
1	Entire table

This argument corresponds to recording table column movements with the ruler or through the Row Height dialog box. If the ruler is used, then all rows are formatted regardless of the selection. If the dialog is used when recording macros, then the format affects only the selected rows.

.LeftIndent	The distance from the left edge of the text and the left margin.

.LineSpacingRule	**The rule for determining the height of the row:**
0	Auto
1	At Least
2	Exactly

.LineSpacing	The height of the row, in points.

.Alignment	**The alignment of the selected rows:**
0	Left
1	Center
2	Right

.PrevRow	Selects the previous row for formatting.
.NextRow	Selects the next row for formatting.

Example

```
TableRowHeight .RulerStyle = "0", .LeftIndent = "1 in", \
.LineSpacingRule = 1, .LineSpacing = "2 li", .Alignment = 0
```

Sets for selected rows a minimum row height of 2 lines and indents the rows to 1 inch from the left margin.

_____
See also
Formatting Statements and Functions
Table Statements and Functions

TableSelectColumn

TableSelectColumn
Selects the column containing the insertion point in a table.

_____
See also
Selection Statements and Functions
Table Statements and Functions

TableSelectRow

TableSelectRow
Selects the row containing the insertion in a table.

_____
See also
Selection Statements and Functions
Table Statements and Functions

TableSelectTable

TableSelectTable
Selects the table containing the insertion point.

_____
See also
Selection Statements and Functions
Table Statements and Functions

TableSplit

TableSplit
Inserts an empty paragraph above the current row in the table; useful for inserting an empty paragraph above a table when the table is the first object in the document.

_____
See also
Editing Statements and Functions
Table Statements and Functions

TableSplitCells

TableSplitCells
Splits previously merged table cells

_____
See also
Editing Statements and Functions
Table Statements and Functions

TableToText

TableToText .ConvertTo = number
Corresponds to Convert To Text dialog box (Table menu); converts the selected cells to normal text; entire rows must be selected.

.ConvertTo	**Corresponds to the Separate Text With group; determines the character used to separate the contents of each cell:**
0	Paragraph Marks
1 or omitted	Tabs (rows ending in paragraph marks)
2	Commas (rows ending in paragraph marks)

Example

TableToText
Converts the selected cells to a tabbed table.

_____
See also
Editing Statements and Functions
Table Statements and Functions

TabType()

n = TabType(Pos)
Returns the alignment of the tab stop at the specified position in the selected text.

Pos	**The position of the tab stop about which information is desired, in points.The function returns:**
-1	If more than one paragraph is selected and all the tabs do not match, or there is no tab stop at that position.

If the tabs match, the tab type is returned, as follows:

0	Left-aligned
1	Centered
2	Right-aligned
3	Decimal

_____
See also
Formatting Statements and Functions

Text

Text x, y, dx, dy, Text$
Creates a rectangle containing static text in a dialog box. A Text instruction must precede the dialog box control with which it is associated.

x, y	The coordinates of the upper-left corner of the rectangle containing the text, in units of 1/8th (for x) and 1/12th (for y) of the system font.
dx, dy	The width and height of the rectangle, in units of 1/8th (for dx) and 1/12th (for dy) of the system font.
Text$	The text to appear in the dialog box. An ampersand (&) preceding a character makes it the keyboard equivalent to selecting in the dialog box. For example, &Programs results in the word Program with an underlined P. when the instruction is executed.

For more information, see "Using Macros" in the Microsoft Word User's Guide.

Examples
Text 7, 10, 21, 9, "Format &Options:"
Results in the text Format Options with the capital O in Options underlined.

_____
See also
Dialog Box Statements and Functions

TextBox

TextBox x, y, dx, dy, .Field
Creates a text box, into which the user can enter information.

x, y	The coordinates of the upper-left corner of the rectangle containing the text, in units of 1/8th (for x) and 1/12th (for y) of the system font.
dx, dy	The width and height of the rectangle, in units of 1/8th (for dx) and 1/12th (for dy) of the system font.
.Field	The text entered in the text box is returned through this argument.
	For more information, see "Using Macros" in the Microsoft Word User's Guide.

Example
TextBox 32, 8, 65, 12, .DataEntered

_____
See also
Dialog Box Statements and Functions

TextToTable

TextToTable
Converts the selected paragraphs to a table.

_____
See also
Table Statements and Functions

Time$()

a$ = Time$()
Returns the current time in the default format specified in the WIN.INI file; if the format isn't set, then the time is returned in the default format for Microsoft Windows.

_____
See also
Standard Basic Statements and Functions

ToggleFieldDisplay

ToggleFieldDisplay
Toggles the display of the selection between field codes and field results.

_____
See also
Field Statements and Functions
View Statements and Functions

TogglePortrait

TogglePortrait
Toggles between portrait and landscape modes.

_____
See also
Tools Statements and Functions
View Statements and Functions

ToggleScribbleMode

ToggleScribbleMode
Toggles hand annotation mode on and off.

_____
See also
Environment Statements and Functions

ToolsBulletListDefault

ToolsBulletListDefault
Corresponds to Bulleted List button on the Toolbar; adds bullets to the selected paragraphs according to the options set in the Bullets And Numbering dialog box (Tools menu).

_____
See also
Formatting Statements and Functions

ToolsBulletsNumbers

ToolsBulletsNumbers [.Type = number,] [.Replace = number,] [.Hang = number,] [.Indent = text,] [.FormatOutline = text,] [.AutoUpdate = number,] [.FormatNumber = number,] [.Punctuation = text,] [.StartAt = text,] [.CharNum = text or number,] [.Font = text,] [.Points = text or number,] [.Remove]
Corresponds to options set in the three parts of the Bullets And Numbering dialog box (Tools menu); sets formats for numbered, bulleted, and outline-numbered paragraphs; not every argument applies to each type of list.

.Type	The category of list to create:
0	Bulleted list.
1	Numbered list.
2	Outline-numbered list.

.Replace	Corresponds to the Replace Only Numbers option in the Numbered List category of the Bullets And Numbering dialog box.
.Hang	If nonzero, sets a hanging indent for the list.
.Indent	The amount of the indent.
.FormatOutline	A format for numbering outlines, corresponding to options listed in the Format list box.
.AutoUpdate	Corresponds to the Auto Update option in the Outline category of the dialog box.
.FormatNumber	A format for numbering lists, corresponding to options listed in the Number Format combo box.
.Punctuation	The separator character that separates numbers from numbered items in numbered lists.
.StartAt	The starting number or letter for the list.
.CharNum	The character or ANSI code for the character to use as the bullet.
.Font	The font for the number in numbered lists.
.Points	The size of the bullet, in points.
.Remove	Corresponds to clicking the Remove button in the dialog box.

Example

```
ToolsBulletsNumbers .Type = 0, .Replace = 0, .Hang = 1, \
.Indent = "0.25 in", .CharNum = "183", .Font = "Symbol", .Points = "10"
```

Formats the selection as a bulleted list, with the bullet defined as character code 183 in the Symbol font, at 10 points in size.

_____
See also
Formatting Statements and Functions
Tools Statements and Functions

ToolsCalculate

ToolsCalculate
n = ToolsCalculate([Expression$])
Corresponds to the Calculate command on the Tools menu; the selection is evaluated as a mathematical expression. See Math Calculations and Equations for instructions on creating expressions. The result of the evaluation is placed on the Clipboard and displayed on the status bar.
The function form evaluates Expression$. If Expression$ is given, this function is equivalent to the = field. Values in Expression$ can be table cell references. If Expression$ is omitted, performs the same operation as the ToolsCalculate statement, but returns the result rather than placing it on the Clipboard.

_____
See also
Tools Statements and Functions

ToolsCompareVersions

ToolsCompareVersions .Name = text
Corresponds to the Compare Versions dialog box (Tools menu); compares the active document with the specified document.

.Name	The name of the document against which the active document is compared.

Example

```
ToolsCompareVersions .Name = "rev1doc.doc"
```

_____
See also
Tools Statements and Functions

ToolsCreateEnvelope

ToolsCreateEnvelope .EnvAddress = text, .EnvReturn = text, .EnvPaperSize = number, .EnvOmitReturn = number, .PrintEnvelope, .AddToDocument
Creates an envelope that prints with the active document.

.EnvAddress	Text specifying the recipient's address.
.EnvReturn	Text specifying the return address.
.EnvPaperSize	Number corresponding to a size listed in the Create Envelope dialog box.

.EnvOmitReturn	**Specifies whether or not to omit the return address:**
0	The return address is not omitted.
-1	The return address is omitted.

Either of the following arguments may be used, but not both.

.PrintEnvelope	Prints the envelope.
.AddToDocument	Adds the envelope to the document.

Example

```
rtn$ = Chr$(13)+Chr$(10)
HerAddr$ = "Jane Doe"+rtn$+"123 Skye St."+rtn$+"OurTown, WA 98107"+rtn$
HisAddr$ = "John Doe"+rtn$+"456 Erde Lane"+rtn$+"OurTown, WA 98107"+rtn$
ToolsCreateEnvelope .EnvAddress=HerAddr$, .EnvReturn=HisAddr$, .EnvPaperSize=0,\
.EnvOmitReturn=0, .AddToDocument
```

_____
See also
Tools Statements and Functions

ToolsGetSpelling

ToolsGetSpelling FillArray$() [, Word$] [, MainDic$] [, SuppDic$]
n = ToolsGetSpelling(FillArray$() [, Word$] [, MainDic$] [, SuppDic$])
Fills a string array with the words suggested as replacements for a misspelled word. Suggestions are appended in the order they appear in the spelling checker.

FillArray$	The string array of suggested replacements.
Word$	The word for which you want suggested replacements. If Word$ is omitted, Word uses the word closest to the insertion point.

MainDic$	**The language of the main dictionary, spelled in that language.**
Language	Spelling
Danish	Dansk
German	Deutsch
English (AUS)	English (AUS)
English (UK)	English (UK)
English (US	English (US)
Spanish	Español
French	Français
French	Français canadien
Italian	Italiano
Dutch	Nederlands
Norwegian Bokmål	Norsk Bokmål
Norwegian Nynorsk	Norsk Nynorsk
Brazilian Portuguese	Português (BR)
Portuguese	Português (POR)
Finnish	Suomi
Swedish	Svenska
SuppDic$	The supplemental dictionary.

The function form returns the number of replacements suggested by the spelling checker. If the word is spelled correctly, 0 (zero) is returned.

Example

```
Sub MAIN
  Dim Spell$(10)                          'Set up array to hold spellings
  ToolsGetSpelling Spell$(), "color"      'Load spellings for color into array
  For Count = 1 To 10                     'Start loop Count
    MsgBox Spell$(Count)                  'Display spelling number Count
  Next                                    'End loop Count
End SUB
```

_____
See also

ToolsGetSynonyms

ToolsGetSynonyms FillArray$() [, Word$]
n = ToolsGetSynonyms(FillArray$() [, Word$])
Fills a string array with all available synonyms for a word.

FillArray$	The string array of available synonyms.
Word$	The word for which you want all available synonyms. If Word$ is omitted, Word uses the word nearest the insertion point.

The function form returns:

0 (zero)	If there are no synonyms available.
-1	If one or more synonyms are available.

_____

See also
Tools Statements and Functions

ToolsGrammar

ToolsGrammar
Corresponds to the Grammar command on the Tools menu; checks grammar in the active document.

_____

See also
Tools Statements and Functions

ToolsHyphenation

ToolsHyphenation [HyphenateCaps = number,] [Confirm = number,] [HotZone = value]
Corresponds to the Hyphenate dialog box (Tools menu); hyphenates the selected text.

HyphenateCaps	If nonzero, sets the Hyphenate Caps option.
Confirm	If nonzero, sets the Confirm option.
HotZone$	Measurement for hyphenation hot zone.

Example

ToolsHyphenation .HyphenateCaps = 1, .Confirm = 1, .HotZone$ = "24 pt"
Sets the Hyphenate Caps and Confirm options, and sets the hot zone to 24 points.

_____

See also
Tools Statements and Functions

ToolsMacro

ToolsMacro [.Name = text,] [.Run,] [.Edit,] [.Delete,] [.Rename,] [.SetDesc,] [.Show = number,] [.Description = text,] [.NewName = text]
Corresponds to the Macro dialog box (Tools menu); runs, creates, deletes, or revises a macro.

.Name	The name of the macro.
.Run	Runs the macro.
.Edit	Opens a macro editing window containing the macro.
.Delete	Deletes the macro.
.Rename	Renames the macro.
.SetDesc	Sets a new description for the macro.

.Show	**Specifies the context:**
Omitted	Word looks for the macro first in the document template, then in NORMAL.DOT, and finally in built-in commands.
0	Commands.
1	Global macros.
2	Template macros.

.Description	Used with .SetDesc, specifies a new description for the macro.
.NewName	Used with .Rename, specifies a new name for the macro.

Example

ToolsMacro .Name = "test", .Show = 1, .Edit
Opens a macro editing window for the global macro Test.

_____

See also
Macro Statements and Functions

ToolsNumberListDefault

ToolsNumberListDefault
Corresponds to the Numbered List button on the Toolbar; adds numbers to the selected paragraphs according to the options set in the Bullets And Numbering dialog box (Tools menu).

_____
See also
Formatting Statements and Functions

ToolsOptions

ToolsOptions .Category = number
Corresponds to the Options dialog box (Tools menu); brings up the Options dialog box for the specified category.

.Category	The category of options to set, as listed in the Options dialog box, starting at 0.

Example
ToolsOptions .Category = 0
Brings up the View category in the Options dialog box.

_____
See also
Environment Statements and Functions
Tools Statements and Functions

ToolsOptionsGeneral

ToolsOptionsGeneral [.Pagination = number,] [.ReplaceSelection = number,] [.DragAndDrop = number,] [.ConfirmConversions = number,] [.InsForPaste = number,] [.Overtype = number,] [.WPHelp = number,] [.WPDocNavKeys,] [.Units = number,] [.ButtonFieldClicks = number]
Corresponds to the General category in the Options dialog box (Tools menu); sets general options in the Word environment.

.Pagination	If nonzero, sets Background Repagination.
.ReplaceSelection	If nonzero, sets Typing Replaces Selection.
.DragAndDrop	If nonzero, enables Drag-And-Drop Text Editing.
.ConfirmConversions	If nonzero, sets Confirm File Conversions.
.InsForPaste	If nonzero, sets Use The INS Key For Paste.
.Overtype	If nonzero, sets Overtype Mode.
.WPHelp	If nonzero, enables WordPerfect Help.
.WPDocNavKeys	If nonzero, enables WordPerfect Document Navigation Keys.

.Units	**Sets the default unit:**
0	Inches
1	Centimeters
2	Points
3	Picas

.ButtonFieldClicks	Sets the number of clicks (1 or 2) required to run a macro with a Macro Button field.

Example
ToolsOptionsGeneral .DragAndDrop = 0
Clears the Drag-And-Drop Text Editing check box.

_____
See also
Tools Statements and Functions

ToolsOptionsGrammar

ToolsOptionsGrammar .Options = number [,.ShowStatistics = number]
Corresponds to the Grammar category in the Options dialog box (Tools menu); sets current grammar options.

.Options	**Corresponds to the use Grammar and Style Rules group:**
0	Strictly (All Rules)
1	For Business Writing
2	For Casual Writing

.ShowStatistics	Corresponds to the Show Readability Statistics After Proofing check box; a on zero value sets the option.

Example
ToolsOptionsGrammar .Options = 1, .ShowStatistics = 1

_____
See also
Tools Statements and Functions

ToolsOptionsKeyboard

ToolsOptionsKeyboard [.Name = text] [,.KeyCode = number] [,.Add] [,.Delete] [,.Show = number] [,.ResetAll] [,.Context = number]
Corresponds to the Keyboard category in the Options dialog box (Tools menu); establishes and modifies the assignment of shortcut key sequences to built-in commands and macros.

.Name	The name of a built-in command or macro.
.KeyCode	A number representing the key sequence, as listed in the following table. The numbers are not equivalent to those in the SendKeys syntax.
.Add	Adds the key assignment.
.Delete	Deletes the key assignment.

.Show	**The type of command to be assigned:**
0	Built-in commands
1	Macros

.ResetAll	Resets all shortcut key assignments to their defaults.

.Context	**Determines where new key assignments are stored.**
0	Global (NORMAL.DOT)
1 or omitted	Active template

Note:	You can return a single key combination to its default function by including the .Delete argument for a keycode that is no longer assigned to any macro or built-in command.

Add this	**For this key**
256	CTRL+
512	SHIFT+
1024	ALT+

Key code	**Produces**
8	BACKSPACE
9	TAB
12	5 on numeric keypad when NumLock is off
13	ENTER
27	ESC
32	SPACEBAR
33	PAGE UP
34	PAGE DOWN
35	END
36	HOME
45	INS
46	DEL
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U

86	V
87	W
88	X
89	Y
90	Z

96	0 on numeric keypad
97	1 on numeric keypad
98	2 on numeric keypad
99	3 on numeric keypad
100	4 on numeric keypad
101	5 on numeric keypad
102	6 on numeric keypad
103	7 on numeric keypad
104	8 on numeric keypad
105	9 on numeric keypad
106	* on numeric keypad
107	+ on numeric keypad
108	' on numeric keypad
109	- on numeric keypad
110	. on numeric keypad
111	/ on numeric keypad
112	F1
113	F2
114	F3
115	F4
116	F5
117	F6
118	F7
119	F8
120	F9
121	F10
122	F11 (ALT+F11 = ALT+F1; Windows limitation)
123	F12 (ALT+F12 = ALT+F2; Windows limitation)
124	F13 (ALT+F13 = ALT+F3; Windows limitation)
125	F14 (ALT+F14 = ALT+F4; Windows limitation)
126	F15 (ALT+F15 = ALT+F5; Windows limitation)
127	F16 (ALT+F16 = ALT+F6; Windows limitation)

Examples

ToolsOptionsKeyboard .Keycode = 322, .Delete
Unassigns CTRL+B from its current assignment.
ToolsOptionsKeyboard .Name = "Bold", .Keycode = 322, .Add
Assigns the bold format to CTRL+B.

_____

See also
Tools Statements and Functions
CountKeys()
KeyCode()
KeyMacro$()

ToolsOptionsMenus

ToolsOptionsMenus .Menu = text, .MenuText = text, .Name = text, [.Add,] [.Delete,] .Show = number, .ResetAll [,.Context = number]
Corresponds to the Menus category in the Options dialog box (Tools menu); establishes and modifies the assignment of menu positions to built-in commands and macros.

.Menu	The menu from which to add or remove the command: &File, &Edit, &View, &Insert, &Format, T&ools, T&able, or &Window. Menu$ must match the menu name in the menu bar, with & in front of the underlined character.
.MenuText	The text as it will appear on the menu with "&" in front of the underlined letter.
.Name	The name of the built-in command or macro associated with the command.
.Add	Adds the command.
.Delete	Deletes the command.

.Show	**The type of command to be assigned:**
0	Built-in commands
1	Macros

.ResetAll	Resets all menu assignments to their defaults.

.Context	**Determines where new menu assignments are stored:**
0	Global (NORMAL.DOT)
1	Active template

Example

ToolsOptionsMenus .Menu = "&Help", .MenuText = "&Test", \
.Name = "TestMacro", .Add, .Show = 1

_____

See also

Tools Statements and Functions
CountMenuItems()
MenuMacro$()
MenuText$()

ToolsOptionsPrint

ToolsOptionsPrint [.Draft = number,] [.Reverse = number,] [.UpdateFields = number,] [.Summary = number,] [.ShowCodes = number,] [.Annotations = number,] [.ShowHidden = number,] [.EnvFeederInstalled = number,] [.WidowControl = number,] [.DfltTrueType = number]
Corresponds to the Print category in the Options dialog box (Tools menu); sets options for printing a document.

.Draft	If 1, sets the Draft Output option.
.Reverse	If 1, sets the Reverse Print Order option.
.UpdateFields	If 1, sets the Update Fields option.
.Summary	If 1, sets the Summary Info option.
.ShowCodes	If 1, sets the Field Codes option.
.Annotations	If 1, sets the Annotations option.
.ShowHidden	If 1, sets the Hidden Text option.
.EnvFeederInstalled	If 1, sets the Printer's Envelope Feeder Has Been Installed option.
.WidowControl	If 1, sets the Widow/Orphan Control option.
.DfltTrueType	If 1, sets the Use TrueType Fonts As Defaults option.

Example

ToolsOptionsPrint .Reverse = 1
Specifies that documents be printed in reverse page order.

_____
See also
Tools Statements and Functions

ToolsOptionsSave

ToolsOptionsSave [.CreateBackup = number,] [.FastSaves = number,] [.SummaryPrompt = number,] [.AutoSave = number,] [.SaveInterval = text]
Corresponds to the Save category in the Options dialog box (Tools menu); sets options for saving documents.

.CreateBackup	If 1, sets the Always Create Backup option; clears the Allow Fast Saves option.
.FastSaves	If 1, sets the Allow Fast Saves option.
.SummaryPrompt	If 1, sets the Prompt For Summary Info option.
.AutoSave	If 1, sets the Automatic Save option.
.SaveInterval	Specifies the time interval for saving documents automatically, in minutes; available only if .AutoSave has been set to 1.

Example

ToolsOptionsSave .AutoSave = 1, .SaveInterval = "10"

_____
See also
Tools Statements and Functions

ToolsOptionsSpelling

ToolsOptionsSpelling [.IgnoreAllCaps = number,] [.IgnoreMixedDigits = number,] [.AlwaysSuggest = number,] [.Type = number,] [.CustomDict1 = text,] [.CustomDict2 = text,] [.CustomDict3 = text,] [.CustomDict4 = text]
Corresponds to the Spelling category in the Options dialog box (Tools menu); sets options for performing spelling checks on a document.

.IgnoreAllCaps	If 1, sets the Ignore Words In Uppercase option.
.IgnoreMixedDigits	If 1, sets the Ignore Words With Numbers option.
.AlwaysSuggest	If 1, sets the Always Suggest option.

.Type	**Type of dictionary being searched:**
0	Normal
1	Concise
2	Complete
3	Medical
4	Legal

.CustomDict1, .CustomDict2, .CustomDict3, .CustomDict4
The names of custom dictionaries to create or add.

Example

ToolsOptionsSpelling .IgnoreAllCaps = 0, .IgnoreMixedDigits = 0,\
.Type = number, .CustomDict1 = "legal.dic"
Specifies that words in all caps and text composed of mixed text and digits not be checked and to use the "legal.dic" custom dictionary.

_____

See also
Tools Statements and Functions

ToolsOptionsToolbar

ToolsOptionsToolbar [.Context = number] [.Tool = number,] [.Button = number,] [.Macro = text,] [.Show = number,] [.ResetAll,] [.ResetTool]
Corresponds to the Toolbar category in the Options dialog box (Tools menu); sets or removes assignment of built-in commands and macros to buttons displayed on the Toolbar.

.Context	**Determines where new menu assignments are stored.**
0	Global (NORMAL.DOT)
1	Active template

.Tool	The number of the tool to change (0 through 29), as listed in the Tool To Change box.
.Button	The number of the button (0 through 22) to assign to the position specified as .Tool, as listed in the Button box. Note: To assign a blank space, set .Button to -1.
.Macro	The name of the built-in command or macro to assign to a button.

.Show	**The type of command to be assigned:**
0	Built-in commands
1	Macros

.ResetAll	Resets all Toolbar assignments to their defaults.
.ResetTool	Resets the specified tool assignment to its default.

Example
ToolsOptionsToolbar .Tool = 29, Button = 57, Macro = "TestMacro", Show = 1
Assigns the happyface button to the button position at the right end of the Toolbar, and assigns the macro TestMacro to the button.

_____
See also
Tools Statements and Functions

ToolsOptionsUserInfo

ToolsOptionsUserInfo [.Name = text,] [.Initials = text,] [.Address = text]
Corresponds to the User Info category in the Options dialog box (Tools menu); changes user identification assignments.

.Name	The name of the current user.
.Initials	The initials of the current user.
.Address	The mailing address of the current user.

Example
ToolsOptionsUserInfo .Name = "Jane Doe", .Initials = "JD"
Sets the current user name and initials.

_____
See also
Tools Statements and Functions

ToolsOptionsView

ToolsOptionsView [.HScroll = number,] [.VScroll = number,] [.StatusBar = number,] [.StyleAreaWidth = text,] [.TableGridlines = number,] [.TextBoundaries = number,] [.PicturePlaceHolders = number,] [.FieldCodes = number,] [.Linebreaks = number,] [.Tabs = number,] [.Spaces = number,] [.Paras = number,] [.Hyphens = number,] [.Hidden = number,] [.ShowAll = number]
Corresponds to the View category in the Options dialog box (Tools menu); displays or hides various elements in documents and the Word environment.

.HScroll	If 1, displays horizontal scroll bars in document windows.
.VScroll	If 1, displays vertical scroll bars in document windows.
.StatusBar	If 1, displays the status bar.
.StyleAreaWidth	Sets the width of the style area, in points, or a measurement in the form of text.
.TableGridlines	If 1, displays table gridlines.
.TextBoundaries	If 1, displays text boundaries.
.PicturePlaceHolders	If 1, displays picture placeholders.
.FieldCodes	If 1, displays field codes.
.Linebreaks	If 1, displays line breaks and fonts as they appear when printed.
.Tabs	If 1, displays tab characters.
.Spaces	If 1, displays space characters.
.Paras	If 1, displays paragraph marks.
.Hyphens	If 1, displays optional hyphens.
.Hidden	If 1, displays hidden text.
.ShowAll	If 1, displays all nonprinting characters.

Example

ToolsOptionsView .Hidden = 1
Displays hidden text in the document.

_____

See also
Tools Statements and Functions

ToolsOptionsWinini

ToolsOptionsWinini .Application = text, .Option = text, .Setting = text [,.Delete] [,.Set]
Corresponds to the WIN.INI category of the Options dialog box (Tools menu); changes options in the Microsoft Windows initialization (WIN.INI) file.

.Application	The name of an application, as listed in the Application box.
.Option	The startup option to modify.
.Setting	The startup option's setting.
.Delete	Deletes the option.
.Set	Sets the option.

Example

ToolsOptionsWinini .Application = "Microsoft Word 2.0", .Option = "DOC-path", .Setting = "c:\worddocs", .Set
Sets the default directory for Word documents to C:\WORDDOCS.

_____
See also
Tools Statements and Functions

ToolsRecordMacro

ToolsRecordMacro [.Name = text,] [.Description = text,] [.Context]
Corresponds to the Record Macro dialog box (Tools menu); records subsequent actions in the specified macro.

.Name	The name of the macro to record; if not given, the next default recording name (Macron) is used.
.Description	Text that describes the macro and appears in the status bar if the macro is assigned to a menu.

.Context	**Where to store the new macro:**
Omitted	Use settings in the Template dialog box (File menu)
0	Global (in NORMAL.DOT)
1	In the active document's template.

Example

ToolsRecordMacro .Name = "TestMacro", .Description = \
"My experimental macro.", .Context = 1
Records a macro called TestMacro and associates it with NORMAL.DOT. "My experimental macro" appears in the status bar if the macro is assigned to a menu and selected.

_____
See also
Tools Statements and Functions

ToolsRepaginateNow

ToolsRepaginateNow
Corresponds to the Tools Repaginate Now command; forces repagination of the entire document.

_____
See also
Formatting Statements and Functions
Tools Statements and Functions

ToolsRevisionMarks

ToolsRevisionMarks [.MarkRevisions = number,] [.RevisionBar = number,] [.NewText = number,] [.Search,] [.AcceptRevisions,] [.UndoRevisions]
Corresponds to the Revision Marks dialog box (Tools menu).

.MarkRevisions	If nonzero, sets the Mark Revisions option.

.RevisionBar	**Specifies type of revision bar:**
0 or omitted	None
1	Left
2	Right
3	Outside

.NewText	**Specifies how to mark new text:**
0 or omitted	None
1	Bold
2	Italic
3	Underline
4	Double underline

.Search	Corresponds to clicking the Search button; searches for next text having revision marks.
.AcceptRevisions	Corresponds to clicking the Accept Revisions button; accepts the current revisions.
.UndoRevisions	Corresponds to clicking the Undo Revisions button; undoes the current revisions.

Example

ToolsRevisionMarks .MarkRevisions = 1, .RevisionBar = 3, .NewText = 3, .AcceptRevisions
Accepts the current revisions, and specifies that revisions be marked with an outside bar and underlining.

_____

See also

Tools Statements and Functions

ToolsSorting

ToolsSorting [.Order = number,] [.Type = number,] [.Separator = number,] [.FieldNum = text,] [.SortColumn = number,] [.CaseSensitive = number]
Corresponds to the Sorting dialog box (Tools menu); sorts the selected paragraphs or rows in a table.

.Order	**Sorting order:**
0 (zero)	Ascending
1	Descending

.Type	**Sort type:**
0 (zero)	Alphanumeric
1	Numeric
2	Date

.Separator	**The type of separator:**
0 (zero)	Comma
1	Tab

.FieldNum	Field number (column) to use as a sort key.
.SortColumn	Corresponds to the Sort Column Only check box; requires column selection.
.CaseSensitive	If nonzero, sets case-sensitive sorting.

Example
ToolsSorting .Order = 0, .Type = 1, .Separator = 1
Sorts the selection in ascending numeric order, specifying that a tab character separate the number from text in each paragraph.

_____
See also
Tools Statements and Functions

ToolsSpelling

ToolsSpelling
Corresponds to the Spelling dialog box (Tools menu); checks spelling in the active document.

_____
See also
Tools Statements and Functions

ToolsSpellSelection

ToolsSpellSelection
Checks the selection. If the selection is only part of a word, or if the insertion point is at the end of a word, the selection is expanded to include the whole word. If the insertion point is not in a word, the next word is checked.

_____
See also
Tools Statements and Functions

ToolsThesaurus

ToolsThesaurus
Corresponds to the Thesaurus command on the Tools menu; lists alternative words for the selection.

_____
See also
Tools Statements and Functions

UCase$()

a$ = UCase$(A$)
Returns a$ converted to uppercase.

_____
See also
Standard Basic Statements and Functions

Underline

Underline [On]

n = Underline()
Adds or removes the Underline character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

_____

See also
Formatting Statements and Functions

UnHang

UnHang
Reduces the left indent in a hanging indent to the left by one tab stop. Keeps the first line of the paragraph indented at the current position.

_____

See also
Formatting Statements and Functions

UnIndent

UnIndent
Reduces the indent of the selected paragraphs. The indent is aligned with the previous tab stop of the first paragraph in the selection. Unindent does not change the setting of a first-line indent.

_____

See also
Formatting Statements and Functions

UnLinkFields

UnLinkFields
Converts the selected fields to plain text and uses the last result. If there are no fields in the selection, an error is generated. Certain kinds of fields cannot be unlinked.

Example

```
Sub MAIN                                    'Begin "OutField"
                                                'Verify action with user
  Msg$ = "Are you sure you want to convert all fields to text?"
  Box = MsgBox (Msg$, "OutField", 33)
  If Box = 0 Then Goto Finish               'If "Cancel" selected then end
  EditSelectAll                                 'Select entire document
  UnLinkFields                                  'Convert all fields in doc to text
Finish:
End Sub
```

_____
See also
Field Statements and Functions

UnLockFields

UnLockFields
Unlocks fields in the selection, for updating.

_____
See also
Field Statements and Functions

UnSpike

UnSpike
Empties the Spike glossary and inserts all contents into the document at the selection. For more information, see Using the Spike to collect and move text and graphics.

_____
See also
Editing Statements and Functions
Glossary Statements and Functions

UpdateFields

UpdateFields
Updates the fields in the selection.

_____
See also
Field Statements and Functions

UpdateSource

UpdateSource
Sends changes in the result of an INCLUDE field back to the source document. The source document must be in Word Normal file format.

_____
See also
Field Statements and Functions

Val()

n = Val(A$)
Returns the numeric value of A$. Strings that begin with characters other than a digit return 0 (zero).

Examples

```
Num=Val("10")
Num=Val("10 Apples")
Evaluate to 10.
Num=Val("ten")
Num=Val("Apartment 10")
```

Evaluate to 0.
a$ = InputBox$("How many dozen apples?")
Total = Val(A$) * 12
Print a$ + " dozen equals" + Str$(Total)
If user types 12 in the dialog box, "12 dozen equals 144" appears in the status bar.

_____

See also
Standard Basic Statements and Functions

ViewAnnotations

ViewAnnotations [On]
logical = ViewAnnotations()
Opens or removes the annotations pane.

On	Determines whether or not the annotations pane is displayed:
0 (zero)	Removes the annotations pane.
1	Displays the annotations pane.
omitted	Toggles display.

If no window is open, an error is generated.

The function form returns:

0 (zero)	If annotations view mode is off
-1	If annotations view mode is on

_____

See also
View Statements and Functions

ViewDraft

ViewDraft [On]
logical = ViewDraft()
Changes the editing view to draft mode.

On	Determines whether or not the active document is displayed in draft mode:
0	Turns off draft mode.
1	Turns on draft mode.
omitted	Toggles draft mode.

The function form returns:

0 (zero)	If draft mode is off.
-1	If draft mode is on.

_____

See also
View Statements and Functions

ViewFieldCodes

ViewFieldCodes [On]
logical = ViewFieldCodes()
Corresponds to the Field Codes command on the View menu; displays field codes or field results.

On	**Determines whether or not to display field codes:**
0	Hides field codes.
1	Displays field codes.
omitted	Toggles display.

If no window is open, an error is generated.

The function form returns:

0 (zero)	If field codes are hidden.
-1	If field codes are displayed.

_____

See also
Fields Statements and Functions
View Statements and Functions

ViewFootnotes

ViewFootnotes [On]
logical = ViewFootnotes()
Opens or removes the footnotes pane.

On	**Determines whether or not the footnotes pane is displayed:**
0 (zero)	Removes the footnotes pane.
1	Displays the footnotes pane.
omitted	Toggles display.

If no window is open, an error is generated.

The function form returns:

0 (zero)	If footnotes view mode is off.
-1	If footnotes view mode is on.

_____

See also
View Statements and Functions

ViewHeaderFooter

ViewHeaderFooter [.Type = number,] [.FirstPage = number,] [.OddAndEvenPages = number,] [.HeaderDistance = text,] [.FooterDistance = text]
Corresponds to the Header/Footer dialog box (View menu); opens the header or footer pane for editing.

.Type	**Specifies the type of header or footer, as described in the following table.**	
.FirstPage	**.OddAndEvenPages**	**.Type values**
0	0	0 = Header 1 = Footer
1	0	0 = Header 1 = Footer 2 = First Header 3 = First Footer
0	1	0 = Even Header 1 = Even Footer 2 = Odd Header 3 = Odd Footer
1	1	0 = First Header 1 = First Footer 2 = Even Header 3 = Even Footer 4 = Odd Header 5 = Odd Footer

.FirstPage	Corresponds to Different First Page check box.
.OddAndEvenPages	Corresponds to Different Odd And Even Pages check box.
.HeaderDistance	Distance from the top of the page.
.FooterDistance	Distance from the bottom of the page.

Example

ViewHeaderFooter
Opens the header pane for the active section.

_____
See also
View Statements and Functions

ViewHeaderFooterLink

ViewHeaderFooterLink
Links the header or footer with that in a previous section. This is not possible in the first section of a document.

_____
See also
Editing Statements and Functions
View Statements and Functions

ViewMenus()

n = ViewMenus()
Returns a value that indicates whether or not at least one document is open.

The function returns:

0	If at least one document is open (all menus are available).
1	If no document is open (only the File, Help, and application Control menus are available).

_____
See also
Environment
View Statements and Functions

ViewNormal

ViewNormal
logical = ViewNormal()
Changes the view to normal view. If no window is open, an error is generated.

The function form returns:

0 (zero)	If normal view mode is on.
-1	If normal view mode is off.

_____

See also
View Statements and Functions

ViewOutline

ViewOutline
logical = ViewOutline()
Changes the view to outline view. If no window is open, an error is generated.

The function form returns:

0 (zero)	If outline view is on.
-1	If outline view is off.

_____

See also
View Statements and Functions

ViewPage

ViewPage
logical = ViewPage()
Changes the view to page layout view. If no window is open, an error is generated.

The function form returns:

0	If page layout view is off.
-1	If page layout view is on.

_____

See also
View Statements and Functions

ViewRibbon

ViewRibbon [On]
logical = ViewRibbon()
Displays or removes the ribbon.

On	**Determines whether or not the ribbon is displayed:**
0	Turns off display.
1	Turns on display.
omitted	Toggles display.

The function form returns:

0	If the ribbon is not displayed.
-1	If the ribbon is displayed.

_____

See also
Formatting Statements and Functions
View Statements and Functions

ViewRuler

ViewRuler [On]
logical = ViewRuler()
Displays or removes the ruler.

On	**Determines whether or not the ruler is displayed:**
0	Turns off display.
1	Turns on display.
omitted	Toggles display.

	The function form returns:
0	If the ruler is not displayed.
-1	If the ruler is displayed.

_____

See also
Formatting Statements and Functions
View Statements and Functions

ViewStatusBar

ViewStatusBar [On]
logical = ViewStatusBar()
Displays or removes the status bar.

On	**Determines whether or not the status bar is displayed:**
0	Turns off display.
1	Turns on display.
omitted	Toggles display.

	The function form returns:
0	If the status bar is not displayed.
-1	If the status bar is displayed.

_____

See also
View Statements and Functions

ViewToolbar

ViewToolbar [On]
logical = ViewToolbar()
Displays or removes the Toolbar.

On	**Determines whether or not the Toolbar is displayed:**
0	Turns off display.
1	Turns on display.
omitted	Toggles display.
	The function form returns:
0	If the Toolbar is not displayed.
-1	If the Toolbar is displayed.

_____

See also
View Statements and Functions

ViewZoom

ViewZoom [.CustomZoomPercent = text,] .ZoomPercent = number or text, .BestFit, .FullPage
Corresponds to the Zoom dialog box (View menu); changes the magnification on the active document.

.CustomZoomPercent	Corresponds to the Custom Zoom field; sets the percentage to magnify the current view shown in the Custom field, but does not change the magnification.

Only one of the following arguments should be used.

.ZoomPercent	The percentage to magnify the current view and the view shown in new windows.
.BestFit	Magnifies the view so that the page width fits within the document window.
.FullPage	Magnifies the view so that the entire page fits within the document window.

Example
ViewZoom .CustomZoomPercent = "75%", ZoomPercent = "100%"
Establishes 75% as the value shown in the Custom field of the Zoom dialog box, and sets the current magnification to 100%.

_____
See also
View Statements and Functions

ViewZoom100

ViewZoom100
Scales the view to 100% in normal view.

_____
See also
View Statements and Functions

ViewZoomPageWidth

ViewZoomPageWidth
Scales the view so that the width of the page is visible.

_____
See also
View Statements and Functions

ViewZoomWholePage

ViewZoomWholePage
Scales the view to see the whole page in page layout view.

_____
See also
View Statements and Functions

VLine

VLine [Count]

Scrolls down vertically by Count number of lines; if omitted, one line is the default. A negative value scrolls up.

_____
See also
View Statements and Functions

VPage

VPage [Count]
Scrolls down vertically by Count number of screens; if omitted, one screen is the default. A negative value scrolls up.

_____
See also
View Statements and Functions

VScroll

VScroll Percentage
n = VScroll()
Scrolls vertically to the place in the document that is the specified percentage of the document's length.
The function form returns the current vertical scroll position as a percentage of the document's size.

_____
See also
View Statements and Functions

While...Wend

While condition
...instruction(s)
Wend
Repeats the instructions in the block while condition is true. If condition is initially false, the loop is never executed.

Example
```
Sub Main
  count = 0
  StartOfDocument
  EditFind .Find = "macro", .Direction = 2
  While EditFindFound()
    count = count + 1
    EditFind .Direction = 2
  Wend
  Print "macro was found "; Str$(count); " times"
End Sub
```
Starts at the beginning of the document and finds each instance of the text "macro". If an instance is found, the variable count is incremented and the loop continues. Finally, the number of instances found is displayed in the status bar.

_____
See also
Standard Basic Statements and Functions

Window()

n = Window()
Returns the number of the active window. The number can range from 0 (zero) to the number of open windows. The number corresponds to the number on the Window menu. A value of 0 indicates no window is open.

_____
See also
Window Statements and Functions

Window1

Window1
Activates Window 1. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window2

Window2
Activates Window 2. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window3

Window3
Activates Window 3. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window4

Window4
Activates Window 4. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____

See also
Window Statements and Functions

Window5

Window5
Activates Window 5. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window6

Window6
Activates Window 6. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window7

Window7
Activates Window 7. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window8

Window8
Activates Window 8. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

Window9

Window9
Activates Window 9. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

_____
See also
Window Statements and Functions

WindowArrangeAll

WindowArrangeAll
Arranges all open windows so that windows do not overlap.

_____
See also
Window Statements and Functions

WindowMainDoc

WindowMainDoc
Switches to a print merge main document. If there is no main document attached to the active document, an error is generated.

_____
See also
Merge Statements and Functions
View Statements and Functions

WindowName$()

a$ = WindowName$(n)
Returns the title of the nth open window. The n corresponds to the number on the Window menu. If n is 0 (zero) or not supplied, the name of the active window is returned.

_____
See also
Window Statements and Functions

WindowNewWindow

WindowNewWindow
Corresponds to the New Window command on the Window menu; creates a copy of the active window.

_____
See also
Window Statements and Functions

WindowPane()

n = WindowPane()
Returns 1 if the window is not split or if the top pane of the active window is selected. Returns 3 if the bottom pane is selected.

_____
See also
Window Statements and Functions

WordLeft

WordLeft [Count,] [Select]
logical = WordLeft([Repeat,] [Select])
Moves the insertion point or extends the selection left by the specified number of words.

Count	The number of words to move; if omitted, 1 is assumed.
Select	If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended.
	The function form returns 0 (zero) if the action cannot be performed; for example, if the insertion point is at the beginning of the document.

_____
See also
Selection Statements and Functions

WordRight

WordRight [Repeat,] [Select]
logical = WordRight([Repeat,] [Select])
Moves the insertion point or extends the selection right by the specified number of words.

Count	The number of words to move; if omitted, 1 is assumed.
Select	If 0 (zero) or omitted, the selection is not extended. If nonzero, the selection is extended. The function form returns 0 (zero) if the action cannot be performed; for example, if the insertion point is at the end of the document.

Example
While WordRight(,1) : Wend
Extends the selection one word at a time until the end of the document is reached.

_____
See also
Selection Statements and Functions

WordUnderline

WordUnderline [On]
logical = WordUnderline()
Adds or removes the Word Underline character format from the selected text.

On	**Specifies whether to add or remove the format:**
1	Formats the selection.
0	Removes the format.
omitted	Toggles the format.

The function form returns:

0 (zero)	If none of the selection is in the format.
-1	If part of the selection is in the format or is a mix of formats.
1	If all of the selection is in the format.

_____
See also
Formatting Statements and Functions

Write

Write [#]StreamNumber, Expressions
Writes the expressions to the file associated with StreamNumber, including delimiters, so that the resulting values can be read by a Read instruction.

_____
See also
File I/O Statements and Functions