This document is an abridged version of the Macros sections from the Technical Reference. In your Word for Windows package, you will find a form to fill out and mail in to purchase the Technical Reference.

Macros: Introduction

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.

1089 Part No. 03915
Kit No. 059-050-007

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).

1089 Part No. 03915
Kit No. 059-050-007

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 (name):

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

1089 Part No. 03915
Kit No. 059-050-007

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.

1089 Part No. 03915
Kit No. 059-050-007

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
2
3
4
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
2
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.

Other example macros are available in your EXAMPLES.DOT file.

Macro Programming Concepts

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

1089 Part No. 03915
Kit No. 059-050-007

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
```


1089 Part No. 03915
Kit No. 059-050-007

```
        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,

1089 Part No. 03915
Kit No. 059-050-007

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

1089 Part No. 03915
Kit No. 059-050-007

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

1089 Part No. 03915
Kit No. 059-050-007

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

1089 Part No. 03915
Kit No. 059-050-007

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	

Expressions

Word can evaluate complex numeric and string expressions. In WordBASIC, all relational expressions return -1 if True and 0 (zero) if False. If strings are used with relational operators,

1089 Part No. 03915
Kit No. 059-050-007

the strings are first converted to ASCII values, and the resulting values are used in the comparison. For example, in the expression If "Apple"<"Orange" Then Print "Apple is less than Orange." Word converts the relational expression into a value that represents True or False. The If statement accepts the value and then performs the operation accordingly.

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

Hint: 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.

All expressions are evaluated such that multiplication and division are performed before addition and subtraction. To perform operations in a different order, use parentheses, as shown in the second example that follows:

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

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:

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).

1089 Part No. 03915
Kit No. 059-050-007

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

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.

1089 Part No. 03915
Kit No. 059-050-007

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


ElseIf Sales > 280 Then

Else

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.

1089 Part No. 03915
Kit No. 059-050-007

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
```


1089 Part No. 03915
Kit No. 059-050-007

```
        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
```


1089 Part No. 03915
Kit No. 059-050-007

```
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
```


1089 Part No. 03915
Kit No. 059-050-007

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)

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.

1089 Part No. 03915
Kit No. 059-050-007

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
Dialog dlgrec
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
End Sub
```


1089 Part No. 03915
Kit No. 059-050-007

Special Bookmarks

Some statements can use the following special bookmarks:

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

1089 Part No. 03915
Kit No. 059-050-007

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:

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

1089 Part No. 03915
Kit No. 059-050-007

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:

indicates that the status of the box is unknown.

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.

option buttons.

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:
Keyword form:

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.

1089 Part No. 03915
Kit No. 059-050-007

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.

Utility Statements and Functions

Abs()

~~Syntax:~~ Num = Abs(n)

Returns the unsigned value of n.

Activate

~~Syntax:~~ Activate WindowText$, [PaneNum]

Activates the window whose title bar is specified by WindowText$. If PaneNum is supplied, a value of 1 or 2 activates the top pane and a value of 3 or 4 activates the bottom pane.

AppActivate

~~Syntax:~~ AppActivate WindowText$, [Immediate]

Activates the application whose title bar is specified by WindowText$. If Immediate is 1, Word immediately switches the focus to the other application. If Immediate is 0 (zero), and 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.

AppInfo$()

~~Syntax:~~ A$ = AppInfo$(TypeOfInfo)

Returns information about the state of Word.

1089 Part No. 03915
Kit No. 059-050-007

If TypeOfInfo is	Result is
1	The environment string; for example, “Windows 2.11”
2	The version number of Word; for example, “1.00”
3	Word is in a special mode; for example, CopyText or MoveText mode
4	X position of the Word window, measured from the left of the screen in points
5	Y position of the Word window, measured from the top of the screen in points
6	Width of the current document workspace in points
7	Height of the current document workspace in points
8	Returns —1 if the application is maximized
9	Amount of total conventional memory
10	Amount of total conventional memory available
11	Amount of total expanded memory
12	Amount of total expanded memory available
13	Returns —1 if a math coprocessor is installed
14	Returns —1 if a mouse is present
15	Amount of disk space available

Values are returned as strings. Use Val(AppInfo$(n)) to convert the string to a number, if appropriate.

AppMaximize

~~Syntax:~~ AppMaximize

Zooms the Word window to full screen size.

AppMaximize()

~~Syntax:~~ Log = AppMaximize()

Returns a nonzero value if the window is maximized.

1089 Part No. 03915
Kit No. 059-050-007

AppMinimize

~~Syntax:~~ AppMinimize

Minimizes the Word window to an icon.

AppMinimize()

~~Syntax:~~ Log = AppMinimize()

Returns a nonzero value if the window is minimized.

AppMove

~~Syntax:~~ AppMove XPos, YPos

Moves the Word window to XPos, YPos relative to the top left of the screen. Values are in points.

AppRestore

~~Syntax:~~ AppRestore

Restores the Word window from a maximized/minimized state.

AppSize

~~Syntax:~~ AppSize XPos, YPos

Resizes the Word window. Values are in points.

Asc()

~~Syntax:~~ Num = Asc(A$)

Returns the ANSI character code of the first character in A$.

1089 Part No. 03915
Kit No. 059-050-007

Beep

~~Syntax:~~ Beep [Beeptype]

Causes the computer's speaker to beep. Beeptype is 1, 2, 3, or 4. If Beeptype is omitted, it is assumed to be 1. The exact tone produced will depend on your hardware configuration. A typical use of Beep is to signal the end of a macro.

Bold

~~Syntax:~~ Bold [On]

Without the argument, toggles bold for the entire selection. If On is nonzero, makes the entire selection bold. If On is 0 (zero), removes bold from the entire selection.

Bold()

~~Syntax:~~ Num = Bold()

Returns 0 (zero) if none of the selection is bold, 1 if all of the selection is bold, or -1 if part of the selection is bold.

BookmarkName$()

~~Syntax:~~ A$ = BookmarkName$(Count)

Returns the name of the bookmark. Count must be in the range from 1 to CountBookmarks().

Call

~~Syntax:~~ [Call] Subname [ParameterList]

Transfers control to a subroutine.

Cancel

~~Syntax:~~ Cancel

Terminates a mode such as ColumnSelect and does not perform the action. See "OK," later in

1089 Part No. 03915
Kit No. 059-050-007

this section.

CenterPara

~~Syntax:~~ CenterPara

Centers the currently selected paragraph(s).

CenterPara()

~~Syntax:~~ Num = CenterPara()

Returns 0 (zero) if none of the selected paragraphs are centered, 1 if all of the selected paragraphs are centered, or -1 if more than one kind of paragraph alignment is used.

ChangeCase

~~Syntax:~~ ChangeCase [Type]

Without an argument, alternates the case of the current selection between all lowercase, all caps, and initial caps based on the first two characters of the selection. If Type is 0 (zero), sets the text to all lowercase. If Type is 1, sets the text to all caps. If Type is 2, sets the text to initial caps.

ChangeRulerMode

~~Syntax:~~ ChangeRulerMode

Cycles the ruler between Paragraph, Table, and Document modes.

CharColor

~~Syntax:~~ CharColor Color

Sets the character color of the selection to Color. The color may be one of the following:

Color argument	**Description**
0	Auto (color specified by the Control Panel setting)

1089 Part No. 03915
Kit No. 059-050-007

1	Black
2	Blue
3	Cyan
4	Green
5	Magenta
6	Red
7	Yellow
8	White

CharColor()

~~Syntax:~~ Num = CharColor()

Returns the numbers set by the CharColor statement, or -1 if all the selected text is not the same color. See CharColor.

CharLeft

~~Syntax:~~ CharLeft [Repeat], [Select]

Moves the selection left by Repeat characters. If the repeat argument is omitted, 1 is assumed. If Select is nonzero, the selection is extended to the left or right by Repeat characters.

CharLeft()

~~Syntax:~~ Log = CharLeft([Repeat], [Select])

Moves the selection left by Repeat characters. Returns 0 (zero) if the action cannot be performed.

CharRight

~~Syntax:~~ CharRight [Repeat], [Select]

Moves the selection right by Repeat characters. If the Repeat argument is omitted,

1 is assumed. If Select is nonzero, the selection is extended to the right by Repeat characters.

1089 Part No. 03915
Kit No. 059-050-007

If Select is 0 (zero) or omitted, the selection is not extended.

CharRight()

~~Syntax:~~ Log = CharRight([Repeat], [Select])

Moves the selection right by Repeat characters. Returns 0 (zero) if the action cannot be performed.

ChDir

~~Syntax:~~ ChDir Name$

Changes directories to the one specified by Name$.

Chr$()

~~Syntax:~~ A$ = Chr$(AnsiCode)

Returns the character whose ANSI code is AnsiCode.

Close

~~Syntax:~~ Close [[#]StreamNumber]

Closes the file attached to StreamNumber. If StreamNumber is not supplied, all open files are closed.

ClosePane

~~Syntax:~~ ClosePane

Closes the current window pane. You use this statement to close a pane in a split document, a header/footer pane, a footnote pane, etc. This does not close a document window, only a pane in a window.

1089 Part No. 03915
Kit No. 059-050-007

CloseUpPara

~~Syntax:~~ CloseUpPara

Makes the space before and after the selected paragraph 0 (zero).

CmpBookmarks()

~~Syntax:~~ Num = CmpBookmarks(Bookmark1$, Bookmark2$)

Compares two named bookmarks and returns one of the following values:

Return value	Meaning
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 does not exist

ColumnSelect

~~Syntax:~~ ColumnSelect

Starts the column selection mode. Cancel ends this mode.

1089 Part No. 03915
Kit No. 059-050-007

ControlRun

~~Syntax:~~ ControlRun Application

Equivalent to the Control Run dialog box. Runs an application from the Word Control menu.

Statement	Effect
ControlRun 0	Runs Clipboard
ControlRun 1	Runs Control Panel

CopyBookmark

~~Syntax:~~ CopyBookmark Bookmark1$, Bookmark2$

Sets Bookmark2$ equal to Bookmark1$.

CopyFormat

~~Syntax:~~ CopyFormat

Copies the formatting of the selected text.

CopyText

~~Syntax:~~ CopyText

Copies text. Equivalent to the copy to key (Shift+F2).

CountBookmarks()

~~Syntax:~~ Num = CountBookmarks()

Returns the number of bookmarks you have defined in the document.

CountFiles()

~~Syntax:~~ Num = CountFiles()

Returns the number of names in the file list on the File menu.

1089 Part No. 03915
Kit No. 059-050-007

CountFonts()

~~Syntax:~~ Num = CountFonts()

Returns the number of fonts available with the printer you've selected.

CountGlossaries()

~~Syntax:~~ Num = CountGlossaries([Context])

Returns the number of glossaries defined in the given context. Context can be 0 (zero) for global or 1 for document template. The default is global.

CountMacros()

~~Syntax:~~ Num = CountMacros([Context], [All])

Returns the number of programs defined in the given context. Context can be 0 (zero) for global or 1 for document template. The default is global.

If All is nonzero, built-in macros are included.

CountStyles()

~~Syntax:~~ Num = CountStyles([Context], [All])

Returns the number of styles defined in the given context. Context can be 0 (zero) for the document or 1 for document template. The default is document.

If All is nonzero, built-in styles are included.

CountWindows()

~~Syntax:~~ Num = CountWindows()

Returns the number of windows in the list on the Window menu.

1089 Part No. 03915
Kit No. 059-050-007

Date$()

~~Syntax:~~ A$ = Date$()

Returns today's date.

DDEExecute

~~Syntax:~~ DDEExecute ChanNum, ExecuteString$

Sends an execute message over the channel ChanNum with an ExecuteString$, which is defined by the receiving application. Use the format described under SendKeys to send specific key sequences.

The channel number must have been opened by the DDEInitiate() function.

If the channel is not valid or if the receiving application refuses to execute the instructions, an error is generated.

DDEInitiate()

~~Syntax:~~ ChanNum = DDEInitiate(App$, Topic$)

Opens a DDE channel to an application. App$ is the application name defined by the other application. Topic$ describes something in the application you are accessing, usually the document containing the data you wish to use.

If DDEInitiate() is successful, it returns the number of the open channel. All subsequent DDE macro functions use this number to specify the channel. This function returns 0 (zero) if it fails to open a channel.

DDEPoke

~~Syntax:~~ DDEPoke ChanNum, Item$, Data$

Sends the data to the item specified by Item$ in the application connected to channel ChanNum. ChanNum must have been opened by the DDEInitiate() function. If DDEPoke is unsuccessful, an error is generated.

1089 Part No. 03915
Kit No. 059-050-007

DDERequest$()

~~Syntax:~~ A$ = DDERequest$(ChanNum, Item$)

Requests the information specified by Item$ over the DDE channel specified by ChanNum. ChanNum must have been opened by the DDEInitiate() function. If DDERequest$() is unsuccessful, a null string ("") is returned.

DDERequest$() returns the data in CF_TEXT format. Pictures or text in Rich Text Format cannot be transferred.

DDETerminate

~~Syntax:~~ DDETerminate ChanNum

Closes the channel ChanNum. The channel must have been opened with the DDEInitiate() function.

DDETerminateAll

~~Syntax:~~ DDETerminateAll

Similar to DDETerminate, but closes all open channels.

Declare

~~Syntax:~~ Declare Sub SubName Lib LibName [ParameterList] [Alias ModuleName]

~~Syntax:~~ Declare Function FunctionName Lib LibName [ParameterList] [Alias ModuleName]

Declares an external library function as a subroutine or function inside a macro.

DeleteBackWord

~~Syntax:~~ DeleteBackWord

Deletes the word immediately preceding the selection but does not place it on the Clipboard.

1089 Part No. 03915
Kit No. 059-050-007

DeleteWord

~~Syntax:~~ DeleteWord

Deletes the word immediately following the selection but does not place it on the Clipboard.

Dim

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

Declares a variable's type and allocates storage space for the variable. For more information on the Dim statement, see Macros: Introduction.

DisableInput

~~Syntax:~~ DisableInput [Disable]

Prevents the Esc key from interrupting a macro. The Esc key is enabled by setting Disable to 0 (zero).

Statement	**Effect**
DisableInput 0	Disable inactive
DisableInput 1	Disable active (default)

DocClose

~~Syntax:~~ DocClose [Save]

Closes the current window or pane. If Save is 1, Word saves the document if it has been edited (considered "dirty") since the last save; if Save is 2, Word does not save the document, but closes the window or pane. If Save is 0 or omitted, Word prompts the user to save the document if it has been edited.

DocMaximize

~~Syntax:~~ DocMaximize

Zooms the document window to application window size. If it is already maximized, the

1089 Part No. 03915
Kit No. 059-050-007

screen is displayed in the restored state.

DocMaximize()

~~Syntax:~~ Log = DocMaximize()

Returns -1 if the window is maximized.

DocMove

~~Syntax:~~ DocMove XPos, YPos

Moves the document window to XPos, YPos relative to the top-left corner of the document area. Values are in points.

DocRestore

~~Syntax:~~ DocRestore

Restores the Word window from a maximized state.

DocSize

~~Syntax:~~ DocSize Width, Height

Sizes the document window to Width, Height. Values are in points.

DocSplit

~~Syntax:~~ DocSplit Percentage

Splits the current window at the given percentage height.

DocSplit()

~~Syntax:~~ Num = DocSplit()

Returns the current window split position as a percentage of the window height, or 0 (zero) if

1089 Part No. 03915
Kit No. 059-050-007

the window isn't split.

DoFieldClick

~~Syntax:~~ DoFieldClick

Simulates a mouse button double-click within the GOTOBUTTON and MACROBUTTON fields' prompt. See the full Technical Reference for information on these fields.

DoubleUnderline

~~Syntax:~~ DoubleUnderline [On]

Without the argument, toggles double underlining for the entire selection. If On is 1, Word makes the entire selection double underlined. If On is 0 (zero), Word removes double underlining from the entire selection.

DoubleUnderline()

~~Syntax:~~ Num = DoubleUnderline()

Returns 0 (zero) if none of the selection is double underlined, 1 if all of the selection is double underlined, or -1 if part of the selection is double underlined or more than one kind of underlining is used.

EditClear

~~Syntax:~~ 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. If Count is supplied, deletes the specified number of characters from the right of the insertion point.

If Count is a negative number, deletes to the left of the insertion point.

1089 Part No. 03915
Kit No. 059-050-007

EditCopy

~~Syntax:~~ EditCopy

Equivalent to the Copy command on the Edit menu. Copies the selection to the Clipboard.

EditCut

~~Syntax:~~ EditCut

Equivalent to the Cut command on the Edit menu. The selection is placed on the Clipboard and then deleted.

EditGlossary

~~Syntax:~~ EditGlossary Name$, [Context]

Equivalent to the Edit Glossary dialog box. Used to define, delete, and insert glossary entries. Context can be 0 (zero) for global or 1 for document template. The default is global.

The default action is Insert. You can perform other actions by appending the command button name from the dialog box (Define or Delete) to the statement.

EditGoTo

~~Syntax:~~ EditGoTo Destination$

Equivalent to the Edit Go To dialog box. Destination$ is a bookmark name, a page number or goto string. See the special bookmarks in the preceding section and Moving the Insertion Point in the User's Reference for more information on bookmarks.

EditHeaderFooter

~~Syntax:~~ EditHeaderFooter [Type], [StartingNum[$]], [NumFormat], [HeaderDistance[$]], [FooterDistance[$]], [FirstPage], [OddAndEvenPages]

Equivalent to the Edit Header/Footer dialog box. Opens the header or footer pane for editing.

The arguments correspond to a check box. If the argument is 1, the check box is on. If the

1089 Part No. 03915
Kit No. 059-050-007

argument is 0 (zero), the check box is off.

EditHeaderFooterLink

~~Syntax:~~ EditHeaderFooterLink

Links the header/footer with a previous section. This is not possible in the first section of a document.

EditPaste

~~Syntax:~~ EditPaste

Equivalent to the Paste command on the Edit menu. Copies the contents of the Clipboard to the insertion point.

EditPasteLink

~~Syntax:~~ EditPasteLink [AutoUpdate]

Equivalent to the Edit Paste Link dialog box. Pastes a DDE (Dynamic Data Exchange) field. If AutoUpdate is 1, EditPasteLink pastes a DDEAUTO field.

For information on DDE, see the full Technical Reference.

EditReplace

~~Syntax:~~ EditReplace [Search$], [Replace$], [WholeWord], [MatchCase], [Confirm], [Format]

Equivalent to the Edit Replace dialog box. Replaces the Search$ string with Replace$. If Search$ and Replace$ are not supplied, the strings used in the previous search and/or replace are used.

To replace formatting in addition to, or instead of, text, use EditReplaceChar, EditReplacePara, EditSearchChar, or EditSearchPara first to set up the formatting, then run EditReplace or EditSearch with Format set to 1.

1089 Part No. 03915
Kit No. 059-050-007

EditReplaceChar

~~Syntax:~~ EditReplaceChar [Font$], [Points[$]], [Color], [Bold], [Italic], [SmallCaps], [Hidden], [Underline], [WordUnderline], [DoubleUnderline], [Position[$]], [Spacing[$]]

Dialog box equivalent; defines the character formatting EditReplace uses to format replacement text. See "FormatCharacter" later in this section.

EditReplacePara

~~Syntax:~~ EditReplacePara [Alignment], [LeftIndent[$]], [RightIndent[$]], [FirstIndent[$]], [Before[$]], [After[$]], [LineSpacing[$]], [Style$], [KeepTogether], [KeepWithNext], [Border], [Pattern], [PageBreak], [NoLineNum]

Dialog box equivalent; defines the paragraph formatting EditReplace uses to format replacement text. The arguments specify options available in the Format Paragraph dialog box. See "FormatParagraph," later in this section.

EditSearch

~~Syntax:~~ EditSearch [Search$], [WholeWord], [MatchCase], [Direction], [Format]

Equivalent to the Edit Search dialog box. Searches for the specified Search$.

The arguments correspond to a check box. If the argument is 1, the check box is on. If the argument is 0 (zero), the check box is off.

EditSearchChar

~~Syntax:~~ EditSearchChar [Font$], [Points[$]], [Color], [Bold], [Italic], [SmallCaps], [Hidden], [Underline], [WordUnderline], [DoubleUnderline], [Position[$]], [Spacing[$]]

Dialog box equivalent; defines the character formatting EditSearch and EditReplace use to find formatted text. See "FormatCharacter," later in this section.

1089 Part No. 03915
Kit No. 059-050-007

EditSearchFound()

~~Syntax:~~ Log = EditSearchFound()

Returns -1 if the last EditSearch was successful. Returns 0 (zero) if not.

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

EditSearchPara

~~Syntax:~~ EditSearchPara [Alignment], [LeftIndent[$]], [RightIndent[$]], [FirstIndent[$]], [Before[$]], [After[$]], [LineSpacing[$]], [Style$], [KeepTogether], [KeepWithNext], [Border], [Pattern], [PageBreak], [NoLineNum]

Dialog box equivalent; defines the paragraph formatting EditSearch and EditReplace use to find formatted text. The arguments specify options available in the Format Paragraph dialog box. See "Format Paragraph," later in this section.

EditSelectAll

~~Syntax:~~ EditSelectAll

Selects the entire document.

EditSummaryInfo

~~Syntax:~~ EditSummaryInfo [Title$], [Subject$], [Author$], [Keywords$], [Comments$], [Directory$], [Template$], [CreateDate$], [LastSavedDate$], [LastSavedBy$],

1089 Part No. 03915
Kit No. 059-050-007

[RevisionNumber], [EditTime$], [LastPrintedDate$], [NumPages], [NumWords], [NumChars], [FileName$]

Equivalent to the Edit Summary Info dialog box. Sets the summary information and allows access to the Statistics dialog box. All the options in the Statistics dialog box are read-only except for the total editing time, which can be set with WordBASIC.

You can append the Update command button name to update the summary information.

EditTable

~~Syntax:~~ EditTable [Modify], [ShiftCells]

Equivalent to the Edit Table dialog box. Modify is 0 (zero) for Row, 1 for Column, or 2 for Selection. ShiftCells is 0 for Horizontally or 1 for Vertically. You can delete, merge, or split cells by appending the Delete, MergeCells, or SplitCells command button name.

EditUndo

~~Syntax:~~ EditUndo

Equivalent to the Undo command on the Edit menu. Undoes the last action, if possible. You can undo certain Word actions, such as Cut and Paste. Some actions can't be undone.

EmptyBookmark()

~~Syntax:~~ Log = EmptyBookmark(Name$)

Returns -1 if Name$ is empty (an insertion point), or 0 (zero) if Name$ is not empty.

EndOfColumn

~~Syntax:~~ EndOfColumn [Select]

Moves the insertion point to the bottom cell in the table column. If Select is a nonzero value, the selection is extended.

1089 Part No. 03915
Kit No. 059-050-007

EndOfDocument

~~Syntax:~~ EndOfDocument [Select]

Moves the selection to the end of the document. If Select is a nonzero value, the selection is extended.

EndOfLine

~~Syntax:~~ EndOfLine [Select]

Moves the selection to the end of the line. If Select is a nonzero value, the selection is extended.

EndOfRow

~~Syntax:~~ EndOfRow [Select]

Moves the selection to the end of the last cell in the table row. If Select is a nonzero value, the selection is extended.

EndOfWindow

~~Syntax:~~ EndOfWindow [Select]

Moves the selection to the end of the window. If Select is a nonzero value, the selection is extended.

Eof()

~~Syntax:~~ Log = Eof(StreamNumber)

Returns -1 when the end of the file attached to the stream number has been reached.

Err

~~Syntax:~~ Err

This is a special variable that contains the error code for the most recent error condition.

1089 Part No. 03915
Kit No. 059-050-007

Error

~~Syntax:~~ Error ErrorNumber

Displays a message corresponding to an error situation. ErrorNumber is an error code.

ExistingBookmark()

~~Syntax:~~ ExistingBookmark(Bookmark$)

Returns -1 if Bookmark$ exists, or 0 (zero) if not.

ExpandGlossary

~~Syntax:~~ ExpandGlossary

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

ExtendSelection

~~Syntax:~~ ExtendSelection [Character$]

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

File1

~~Syntax:~~ File1

Equivalent to selecting the first listed file on the File menu. Opens the first file. An error is generated if you attempt to open a nonexistent file slot. For example, you cannot use File4 if only three files are listed under the File menu.

File2

~~Syntax:~~ File2

1089 Part No. 03915
Kit No. 059-050-007

Equivalent to selecting the second listed file on the File menu. Opens the second file. An error is generated if you attempt to open a nonexistent file slot. For example, you cannot use File4 if only three files are listed under the File menu.

File3

~~Syntax:~~ File3

Equivalent to selecting the third listed file on the File menu. Opens the third file. An error is generated if you attempt to open a nonexistent file slot. For example, you cannot use File4 if only three files are listed under the File menu.

File4

~~Syntax:~~ File4

Equivalent to selecting the fourth listed file on the File menu. Opens the fourth file. An error is generated if you attempt to open a nonexistent file slot. For example, you cannot use File4 if only three files are listed under the File menu.

FileClose

~~Syntax:~~ FileClose [Save]

Equivalent to the Close command on the File menu. Closes the current file and associated windows. The Save argument determines whether a save is forced: 1 forces a save, 2 forces no save, 0 prompts the user to save edited documents.

FileExit

~~Syntax:~~ FileExit [Save]

Equivalent to the Exit command on the File menu. Quits Word. If any open documents have been edited and Save is omitted or is 0 (zero), you are prompted to save each changed document. If Save is 1, all edited documents are automatically saved before exiting. If Save is 2, the documents are not saved.

1089 Part No. 03915
Kit No. 059-050-007

FileFind

~~Syntax:~~ FileFind [SortBy], [SearchList$], [Title$], [Subject$], [Author$], [Keywords$], [SavedBy$], [Text$], [DateCreatedFrom$], [DateCreatedTo$], [DateSavedFrom$], [DateSavedTo$], [MatchCase], [SearchAgain]

Equivalent to the File Find dialog box. It can be used to change the search criteria in subsequent FileFind statements. If you record a macro with FileFind, any other actions you perform in the File Find dialog box at that time, such as opening or deleting a document, editing summary information, or printing, are also recorded.

FileName$()

~~Syntax:~~ A$ = FileName$(n)

Returns the nth file in the file list. If n is 0 (zero), the name of the current file is returned. If n is greater than the number of files in the file cache, an error is generated. If there is no current document, FileName$(0) returns an empty string.

FileNew

~~Syntax:~~ FileNew [NewTemplate], [Template$]

Equivalent to the File New dialog box.

FileOpen

~~Syntax:~~ FileOpen Name$, [ReadOnly]

Equivalent to the File Open dialog box. Opens the named document. An error is generated if the document does not exist. If ReadOnly is 1, the document is opened as read-only.

FilePrint

~~Syntax:~~ FilePrint [Type], [NumCopies], [Range], [From$], [To$], [Reverse], [Draft], [UpdateFields], [PaperFeed], [Summary], [Annotations], [ShowHidden], [ShowCodes], [FileName$]

1089 Part No. 03915
Kit No. 059-050-007

Equivalent to the File Print dialog box.

FilePrinterSetup

~~Syntax:~~ FilePrinterSetup [Printer$]

Equivalent to the File Printer Setup dialog box. Printer$ is the name of the new printer to be activated. Enter this argument exactly as it appears in the File Printer Setup dialog box.

The Setup command button name can be appended to display the dialog box showing the printer options.

FilePrintMerge

~~Syntax:~~ FilePrintMerge [From], [To]

Equivalent to the File Print Merge dialog box. If From or To is nonzero, Word merges the specified records only.

The New Document command button name can be appended to direct the output to a new document.

FilePrintPreview

~~Syntax:~~ FilePrintPreview [On]

Equivalent to the Print Preview command on the File menu. Without On, toggles print preview mode. If On is nonzero, turns on print preview mode; if On is 0 (zero), turns off print preview mode.

FilePrintPreviewBoundaries

~~Syntax:~~ FilePrintPreviewBoundaries [On]

Displays the text boundaries if On is nonzero; turns off display if On is 0 (zero). If On is omitted, toggles the display of text boundaries.

1089 Part No. 03915
Kit No. 059-050-007

FilePrintPreviewPages

~~Syntax:~~ FilePrintPreviewPages [Pages]

Without the argument, toggles display between one and two pages.

Pages argument	Description
0	Toggles the display state (default)
1	One page
2	Two pages

FileSave

~~Syntax:~~ FileSave

Equivalent to the Save command on the File menu. Saves the current document.

FileSaveAll

~~Syntax:~~ FileSaveAll [Save]

Equivalent to the Save All command on the File menu. Prompts the user to save all changed files including NORMAL.DOT. If Save is 1, all edited documents are automatically saved. If Save is 2, the documents are not saved. If Save is 0 or omitted, Word prompts the user to save all changed files.

FileSaveAs

~~Syntax:~~ FileSaveAs [Name$], [Format], [FastSave], [CreateBackup], [LockAnnot]

Equivalent to the File Save As dialog box. Saves the current document with a new name and/or format. Name$ specifies the new name. Format specifies the new format.

Format argument	Document type
0	Normal (Word format)
1	Document Template
2	Text Only (extended characters saved in ANSI character set)

1089 Part No. 03915
Kit No. 059-050-007

3	Text+Breaks (plain text with line breaks; extended characters saved in ANSI character set)
4	Text Only (extended characters saved in IBM PC character set)
5	Text+Breaks (text with line breaks; extended characters saved in IBM PC character set)
6	Rich Text Format (RTF)

Other file formats can be specified. They must be listed in your WIN.INI file under the Microsoft Word entry.

Files$()

~~Syntax:~~ A$ = Files$(FileSpec$)

Returns the first filename matching FileSpec$. If FileSpec$ is not supplied, the next file matching the last-used FileSpec$ is returned. This function can be used to get a list of files matching a FileSpec$ by specifying the FileSpec$ on the first iteration, and then omitting it thereafter. If no files match, a null string ("") is returned. Files$ (".") returns the current directory.

Font

~~Syntax:~~ Font Name$, [Size]

Applies the named font to the selection. You can include the Size argument instead of following this statement with the FontSize statement.

Font$()

~~Syntax:~~ A$ = Font$()

~~Syntax:~~ A$ = Font$(Count)

Returns the font name of the current selection. If the selection has more than one font, a null string is returned. If Count is supplied, Font$() returns the name of the font Count. Count must be in the range 1 to CountFonts().

1089 Part No. 03915
Kit No. 059-050-007

FontSize

~~Syntax:~~ FontSize Size

Sets the size of the current selection in points.

FontSize()

~~Syntax:~~ Num = FontSize()

Returns the font size of the current selection. If the selection has more than one font size, 0 (zero) is returned.

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 more information on the For and Next statements, see Macros: Introduction.

FormatCharacter

~~Syntax:~~ FormatCharacter [Font$], [Points[$]], [Color], [Bold], [Italic], [SmallCaps], [Hidden], [Underline], [WordUnderline], [DoubleUnderline], [Position[$]], [Spacing[$]]

Equivalent to the Format Character dialog box. Applies character formatting to the selection. Some arguments take measurements in points. Other arguments correspond to a check box.

FormatDefineStyles

~~Syntax:~~ FormatDefineStyles Name$, [BasedOn$], [NextStyle$], [AddToTemplate], [NewName$], [FileName$], [Source]

Equivalent to the Format Define Styles dialog box. Defines a new style with the specified

1089 Part No. 03915
Kit No. 059-050-007

Name$. If a style with that name already exists, that style is made the current style. FormatDefineStyles sets up the style; to define the character, paragraph, and tab formats, use the FormatDefineStylesChar, FormatDefineStylesPara, and FormatDefineStylesTabs statements described later in this section. To redefine an existing style, include the specific arguments with the FormatDefineStyles statement.

The BasedOn$ argument specifies a style on which to base the new style. The NextStyle$ argument specifies the style to be applied after the new style.

AddToTemplate can be 0 for the document only, or 1 for the document and its template.

The Delete, Rename, and Merge command button names can be appended.

The NewName$ argument specifies a new name for the style; it is used only in conjunction with the Rename command button.

The FileName$ argument is used only with the Merge command button. It specifies the template or document whose style sheet is to be merged with that of the current document or template.

Source is used only in conjunction with the Merge command button name, and can be 0 (zero) (from the current document or template to a specified document or template) or 1 (from a specified document or template to the current document or template).

FormatDefineStylesChar

Dialog box equivalent; defines the current style with the specified character properties. This statement takes the same arguments as its corresponding format function. See "FormatCharacter," earlier in this section.

FormatDefineStylesPara

Dialog box equivalent; defines the current style with the specified paragraph properties. This statement takes the same arguments as its corresponding format function. See "FormatParagraph," later in this section.

1089 Part No. 03915
Kit No. 059-050-007

FormatDefineStylesPosition

Dialog box equivalent; defines the current style with the specified position properties. This statement takes the same arguments as its corresponding format function. See "FormatPosition," later in this section.

FormatDefineStylesTabs

Dialog box equivalent; defines the current style with the specified tab properties. This statement takes the same arguments as its corresponding format function. See "FormatTabs," later in this section.

FormatDocument

~~Syntax:~~ FormatDocument [PageWidth[$]], [PageHeight[$]], [DefTabs[$]], [TopMargin[$]], [BottomMargin[$]], [LeftMargin[$]], [RightMargin[$]], [Gutter[$]], [MirrorMargins], [FootnotesAt], [StartingNum[$]], [RestartNum], [Template$], [WidowControl]

Equivalent to the Format Document dialog box. Applies document formatting properties.

Some arguments take measurements in points. Other arguments correspond to a check box.

To set the global or template default, append the SetDefault command button name to this statement. This is a powerful argument that changes the default document properties to those specified in the statement.

FormatParagraph

~~Syntax:~~ FormatParagraph [Alignment], [LeftIndent[$]], [RightIndent[$]], [FirstIndent[$]], [Before[$]], [After[$]], [LineSpacing[$]], [Style$], [KeepTogether], [KeepWithNext], [Border], [Pattern], [PageBreak], [NoLineNum]

Equivalent to the Format Paragraph dialog box. Applies paragraph formatting.

The LeftIndent[$], RightIndent[$], and FirstIndent[$] arguments specify the amount of left, right, and first-line indents, respectively. The Before[$] and After[$] arguments specify the amount of spacing above and below a paragraph, respectively. The LineSpacing[$] argument

1089 Part No. 03915
Kit No. 059-050-007

specifies the amount of spacing for all lines in a paragraph. The Style$ argument specifies a style to be applied to a paragraph. The KeepTogether and KeepWithNext arguments prevent page breaks within a paragraph and between paragraphs, respectively.

The PageBreak argument inserts a page break before printing the paragraph. The NoLineNum argument turns off line numbering for the paragraph.

FormatPicture

~~Syntax:~~ FormatPicture [Border], [ScaleY], [ScaleX], [CropTop[$]], [CropLeft[$]], [CropBottom[$]], [CropRight[$]]

Equivalent to the Format Picture dialog box. Applies picture formatting properties.

Some arguments take measurements in points. Other arguments correspond to check boxes.

FormatPosition

~~Syntax:~~ FormatPosition [Horizontal[$]], [HRelativeTo], [Vertical[$]], [VRelativeTo], [DistanceFromText], [ParagraphWidth[$]]

Equivalent to the Format Position dialog box. Applies position formatting to the selected paragraphs.

The Reset command button name can be appended to cancel the position formatting of the paragraphs.

FormatSection

~~Syntax:~~ FormatSection [Columns], [ColumnSpacing[$]], [ColLine], [SectionStart], [Footnotes], [LineNum], [StartingNum[$]], [FromText[$]], [CountBy], [NumMode], [VertAlign]

Equivalent to the Format Section dialog box. Applies section formatting properties to the selection. Some arguments take measurements in points or numbers. Other arguments correspond to check boxes.

1089 Part No. 03915
Kit No. 059-050-007

FormatStyles

~~Syntax:~~ FormatStyles Name$, [Create]

Equivalent to the Format Styles dialog box. Applies the style in Name$ to the selected paragraphs. If the style does not exist and Create is not specified or is 0 (zero), an error is generated. If Create is specified as 1, the style is created with the properties of the selection, if it doesn't already exist.

FormatTable

~~Syntax:~~ FormatTable [FromColumn], [Column], [ColumnWidth], [SpaceBetweenCols[$]], [IndentRows[$]], [MinimumRowHeight], [OutlineBorder], [TopBorder], [BottomBorder], [InsideBorder], [LeftBorder], [RightBorder], [AlignRows], [ApplyTo]

Equivalent to the Format Table dialog box. When recording, pressing the Next or Prev Columns command button records a new FormatTable command.

FormatTabs

~~Syntax:~~ FormatTabs [Position], [Align], [Leader]

Equivalent to the Format Tabs dialog box. Position is a measurement in points.

Align argument	**Alignment**
0	Left
1	Centered
2	Right
3	Decimal

Leader argument	**Leader character**
0	None
1	Dot
2	Dash
3	Underline

Set is the default action. You can also clear specified tabs or clear all tabs by appending the Clear or ClearAll command button name to the statement.

1089 Part No. 03915
Kit No. 059-050-007

Function...End Function

Syntax:

Statement(s)

End Function

Defines a function. The ParameterList is a list of variables, separated by commas, for receiving arguments to the function. For more information on user-defined functions, see Macros: Introduction.

GetBookmark$()

Syntax: A$ = GetBookmark$(BookmarkName$)

Returns the text at the specified bookmark.

GetCurValues

Syntax: GetCurValues DialogRecord

Stores in DialogRecord the current values for a previously dimensioned dialog box. For more information, see Macros: Introduction.

GetGlossary$()

Syntax: A$ = GetGlossary$(Name$, [Context])

Returns the text of the glossary entry in Name$. The Context is 0 (zero) for global (default) or 1 for document template.

GetProfileString$()

Syntax: A$ = GetProfileString$([App$], Key$)

Gets a value from the current WIN.INI file. App$ is the name of the Microsoft Windows

1089 Part No. 03915
Kit No. 059-050-007

application. If the application is not specified, the string [Microsoft Word] is used. If the Key$ is not found, the function returns a null string.

GlossaryName$()

~~Syntax:~~ A$ = GlossaryName$(Count, [Context])

Returns the name of the glossary defined in the given context (global or document template). Count must be in the range from 1 to CountGlossaries(Context). The name is taken from the list in the given context. Context is 0 (zero) for global, 1 for document template.

GoBack

~~Syntax:~~ GoBack

Toggles among the last three selections where text or formatting has changed.

Goto

~~Syntax:~~ Goto Label/LineNumber

Branches unconditionally to a label or line number.

GrowFont

~~Syntax:~~ GrowFont

Increases the size of the selected font. Can be used either on the selection, or at the insertion point.

HangingIndent

~~Syntax:~~ HangingIndent

Sets the indent of the selection to the next tab stop in the first paragraph. Sets the first line of the paragraph flush with the left margin.

1089 Part No. 03915
Kit No. 059-050-007

Help

~~Syntax:~~ Help

Activates Help. Equivalent to pressing F1.

HelpAbout

~~Syntax:~~ HelpAbout

Displays a dialog box with the Word version number and copyright information.

HelpActiveWindow

~~Syntax:~~ HelpActiveWindow

Activates Help for the active window.

HelpContext

~~Syntax:~~ HelpContext

Activates context-sensitive Help. Equivalent to pressing Shift+F1.

HelpIndex

~~Syntax:~~ HelpIndex

Displays the list of Help topics.

HelpKeyboard

~~Syntax:~~ HelpKeyboard

Displays list of keyboard Help topics.

HelpTutorial

~~Syntax:~~ HelpTutorial

1089 Part No. 03915
Kit No. 059-050-007

Starts the Tutorial.

HelpUsingHelp

~~Syntax:~~ HelpUsingHelp

Displays Help topics on how to use Help.

Hidden

~~Syntax:~~ Hidden [On]

Without an argument, toggles hidden text for the entire selection. If On is nonzero, makes the entire selection hidden text if the first character is hidden. If On is 0 (zero), removes hidden text from the entire selection.

Hidden()

~~Syntax:~~ Num = Hidden()

Returns 0 (zero) if none of the selection is hidden text, 1 if all of the selection is hidden text, or -1 if part of the selection is hidden text.

HLine

~~Syntax:~~ HLine [Count]

Scrolls horizontally to the right by Count lines. If Count is not specified, one line is the default. "Lines" mean the amount the screen is scrolled by clicking the mouse in a horizontal scroll bar arrow. A negative Count scrolls to the left.

HPage

~~Syntax:~~ HPage [Count]

Scrolls horizontally by Count screens. If Count is not specified, one screen is the default. A negative Count scrolls to the left.

1089 Part No. 03915
Kit No. 059-050-007

HScroll

~~Syntax:~~ HScroll Percentage

Scrolls horizontally the specified percentage of the document width.

HScroll()

~~Syntax:~~ Num = HScroll()

Returns the current horizontal scroll position as a percentage of the document width.

IconBarMode

~~Syntax:~~ IconBarMode

Activates icon bar mode.

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

~~Syntax:~~

~~Syntax:~~

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. For more information, see Macros: Introduction.

1089 Part No. 03915
Kit No. 059-050-007

Indent

~~Syntax:~~ Indent

Indents the selection. The indent is aligned with the next tab stop. Indent does not change a first-line indent.

Input

~~Syntax:~~ Input [#]StreamNumber, Variable, [Variable]

Reads a line from the file specified by #StreamNumber into the variables listed. The line read from the file is separated into individual values by commas. If a StreamNumber is not specified, you are prompted in the status bar.

Input$()

~~Syntax:~~ A$ = Input$(n, StreamNumber)

Reads n characters from the file specified by StreamNumber.

InputBox$()

~~Syntax:~~ A$ = InputBox$(Prompt$, [Title$], [Default$])

Displays an editable dialog box. Returns the text that was in the box when OK was chosen. If you specified a default, it is loaded into the dialog box when it is displayed.

Insert

~~Syntax:~~ Insert Text$

Inserts the given text at the insertion point. Nonprinting characters are inserted as Chr$(n) statements.

Value	**Character inserted**
Chr$(9)	Tab
Chr$(11)	Linefeed
Chr$(30)	Nonbreaking hyphen

1089 Part No. 03915
Kit No. 059-050-007

Chr$(31)	Optional hyphen
Chr$(34)	Quotation marks
Chr$(160)	Nonbreaking space

InsertBookmark

~~Syntax:~~ InsertBookmark Name$

Equivalent to the Insert Bookmark dialog box. Creates or deletes the named bookmark. If the Delete command button is appended, the bookmark is deleted. If Delete is not specified, the bookmark is created at the current selection.

If you specify a nonexistent bookmark for deletion, an error is generated.

InsertBreak

~~Syntax:~~ InsertBreak Type

Equivalent to the Insert Break dialog box. Inserts a page, section, or column break at the current selection.

Type argument	**Break type**
0	Page
1	Column

The following are section breaks:

Type argument	**Break type**
2	Next
3	Continuous
4	Even
5	Odd

1089 Part No. 03915
Kit No. 059-050-007

InsertColumnBreak

~~Syntax:~~ InsertColumnBreak

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.

InsertDateField

~~Syntax:~~ InsertDateField

Inserts a DATE field at the selection.

InsertField

~~Syntax:~~ InsertField Field$

Equivalent to the Insert Field dialog box. Inserts the specified field at the selection. Do not include the field characters in Field$. For more information on inserting fields, see the full Technical Reference.

InsertFieldChars

~~Syntax:~~ InsertFieldChars

Inserts field characters ({ }) at the selection.

InsertFile

~~Syntax:~~ InsertFile Name$, [Range$], [Link]

Equivalent to the Insert File dialog box. Inserts the named file at the current selection.

Range$ refers to a bookmark if Name$ refers to a Word document. If Name$ refers to another document type (for example, a Microsoft Excel worksheet), then Range$ refers to a named range. Only that part of the file is inserted. If Link is 1, a link to the file is inserted instead of the actual file.

1089 Part No. 03915
Kit No. 059-050-007

InsertFootnote

~~Syntax:~~ InsertFootnote [Reference$]

Equivalent to the Insert Footnote dialog box. Inserts a footnote at the current selection.

Reference$ is footnote reference text that you supply.

To insert a footnote separator, continued footnote separator or notice for continued footnotes, append the Separator, ContSeparator, or ContNotice command button name.

InsertIndex

~~Syntax:~~ InsertIndex [Type], [HeadingSeparator], [Replace]

Equivalent to the Insert Index dialog box. Inserts an INDEX field at the current selection.

Type is 0 (zero) for a normal index (default) or 1 for a run-in index. HeadingSeparator is 0 for none (default), 1 for a blank line, or 2 for a letter.

If Replace is 1, the existing index is overwritten. If Replace is omitted or 0, the existing index is not overwritten.

InsertIndexEntry

~~Syntax:~~ InsertIndexEntry [Entry$], [Range$], [Bold], [Italic]

Equivalent to the Insert Index Entry dialog box. Inserts an XE field at the current selection. If Entry$ is omitted, the selection is the entry. The arguments correspond to check boxes.

InsertPageBreak

~~Syntax:~~ InsertPageBreak

Inserts a page break at the current selection.

1089 Part No. 03915
Kit No. 059-050-007

InsertPageField

~~Syntax:~~ InsertPageField

Inserts a PAGE field at the current selection.

InsertPageNumbers

~~Syntax:~~ InsertPageNumbers [Type], [Position]

Dialog box equivalent; inserts a current PAGE field into the header or footer.

Type is 0 (zero) for header, 1 for footer. Position is 0 (left aligned), 1 (centered), or 2 (right aligned).

InsertPara

~~Syntax:~~ InsertPara

Inserts a paragraph mark at the current selection.

InsertPicture

~~Syntax:~~ InsertPicture [Name$]

Equivalent to the Insert Picture dialog box. Inserts an IMPORT field at the current selection. If Name$ is not supplied, a 1-inch graphic frame with a single border is inserted.

InsertTable

~~Syntax:~~ InsertTable [NumColumns], [NumRows], [InitialColWidth[$]], [ConvertFrom]

Equivalent to the Insert Table dialog box. Choosing Format is recorded as an InsertTable statement followed by a FormatTable statement.

InsertTableOfContents

~~Syntax:~~ InsertTableOfContents [Source], [From], [To], [Replace]

1089 Part No. 03915
Kit No. 059-050-007

Equivalent to the Insert Table Of Contents dialog box. Inserts a TOC field at the current selection.

Source is 0 (zero) for outline headings, or 1 for table entry fields. From and To refer to the outline levels used.

If Replace is 1, the existing table of contents is overwritten. If Replace is omitted or 0 (zero), the existing table of contents is not overwritten.

InsertTableToText

~~Syntax:~~ InsertTableToText [ConvertTo]

Dialog box equivalent; converts the selected cells to normal text.

ConvertTo may be 0 (zero) for paragraphs, 1 for tab-delimited text, or 2 for comma-delimited text.

InsertTimeField

~~Syntax:~~ InsertTimeField

Inserts a TIME field at the current selection.

Instr()

~~Syntax:~~ Num = Instr([Index], Source$, Search$)

Searches for Search$ in Source$. Returns the number of the character where Search$ started, or 0 (zero) if Search$ is not found in Source$. If Index is supplied, the search starts at character Index.

Int()

~~Syntax:~~ Num = Int(n)

Returns the integer part of n.

1089 Part No. 03915
Kit No. 059-050-007

IsDirty()

~~Syntax:~~ Log = IsDirty()

Returns -1 if the document has been changed (made dirty) since the last save, 0 (zero) if the document has not been changed.

Italic

~~Syntax:~~ Italic [On]

Without the argument, toggles italic for the entire selection. If On is nonzero, makes the entire selection italic. If On is 0 (zero), removes italic from the entire selection.

Italic()

~~Syntax:~~ Num = Italic()

Returns 0 (zero) if none of the selection is italic, 1 if all of the selection is italic, or -1 if part of the selection is italic.

JustifyPara

~~Syntax:~~ JustifyPara

Justifies the selected paragraphs.

JustifyPara()

~~Syntax:~~ Num = JustifyPara()

Returns 0 (zero) if none of the selected paragraphs are justified, 1 if all of the selected paragraphs are justified, or -1 if more than one kind of paragraph alignment is used.

Kill

~~Syntax:~~ Kill Name$

1089 Part No. 03915
Kit No. 059-050-007

Deletes the file specified by Name$.

LCase$()

~~Syntax:~~ A$ = LCase$(Source$)

Returns Source$ converted to lowercase.

Left$()

~~Syntax:~~ A$ = Left$(Source$, n)

Returns the leftmost n characters of Source$.

LeftPara

~~Syntax:~~ LeftPara

Left aligns the selected paragraphs.

LeftPara()

~~Syntax:~~ Num = LeftPara()

Returns 0 (zero) if none of the selected paragraphs are left aligned, 1 if all of the selected paragraphs are left aligned, or -1 if more than one kind of paragraph alignment is used.

Len()

~~Syntax:~~ Num = Len(Source$)

Returns the number of characters in Source$.

Let

~~Syntax:~~ [Let] Var = Expression

Assigns the value of an expression to a variable. Let is optional.

1089 Part No. 03915
Kit No. 059-050-007

LineDown

~~Syntax:~~ LineDown [Repeat], [Select]

Moves the selection down by Repeat lines. If the Repeat argument is omitted, 1 is assumed. If Select is nonzero, the selection is extended down by Repeat lines.

LineDown()

~~Syntax:~~ Log = LineDown([Repeat], [Select])

Moves the selection down by Repeat lines. Returns 0 (zero) if the action cannot be completed.

Line Input

~~Syntax:~~ Line Input [#]StreamNumber, Variable$

Reads an entire line from the file specified by StreamNumber and puts the result in the specified string variable. If a StreamNumber is not specified, you are prompted in the status bar. Similar to the Input statement but **LineInput** doesn't break the line into separate values at commas.

LineUp

~~Syntax:~~ LineUp [Repeat], [Select]

Moves the selection up by Repeat lines. If the Repeat argument is omitted, 1 is assumed. If Select is nonzero, the selection is extended up by Repeat lines. If Select is 0 (zero) or omitted, the selection is not extended.

LineUp()

~~Syntax:~~ Log = LineUp([Repeat], [Select])

Moves the selection up by Repeat lines. Returns 0 (zero) if the action cannot be completed. For example, the function would return 0 if the insertion point is at the beginning of the document.

1089 Part No. 03915
Kit No. 059-050-007

LockFields

~~Syntax:~~ LockFields

Prevents the fields within the selection from being updated.

Lof()

~~Syntax:~~ Num = Lof(StreamNumber)

Returns the length of the file, in bytes.

MacroAssignToKey

~~Syntax:~~ MacroAssignToKey [Name$], [KeyCode], [Context]

Equivalent to the Macro Assign to Key dialog box. You can assign a macro to any key or key combination.

Assigns the macro Name$ to the specified KeyCode. KeyCode is a number representing the exact key. The number is not equivalent to the SendKeys syntax.

Context is 0 (zero) for global or 1 for document template.

Assign is the default action. The ResetAll command button name can be appended to return the key assignments to the default state. The UnAssign command button name removes a macro connection to a specific key.

Add this	**For this key**
1024	Alt +
512	Shift +
256	Ctrl +

Key Code	**Produces**
8	Backspace
9	Tab
12	5 on numeric keypad when NumLock is off
13	Enter

1089 Part No. 03915
Kit No. 059-050-007

27	Esc
32	Space
33	PgUp
34	PgDn
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

1089 Part No. 03915
Kit No. 059-050-007

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

1089 Part No. 03915
Kit No. 059-050-007

122	F11
123	F12
124	F13
125	F14
126	F15
127	F16

MacroAssignToMenu

~~Syntax:~~ MacroAssignToMenu [Name$], [Menu$], [MenuText$], [Context]

Equivalent to the Macro Assign To Menu dialog box. Assigns the macro Name$ to the specified Menu$ with MenuText$. Menu$ can be File, Edit, View, Insert, Format, Utilities, Macro, or Window.

Context is 0 (zero) for global or 1 for document template.

Assign is the default action. You can append the ResetAll or UnAssign command button name to return the menu assignments to the default state or remove a macro from a menu.

MacroEdit

~~Syntax:~~ MacroEdit Name$, [Context], [Description$], [ShowAll], [NewName$]

Equivalent to the Macro Edit dialog box. Displays the Name$ macro for editing. Context is 0 (zero) for global (default) or 1 for document template. The Description$ refers to the text that appears in the status bar if the macro is assigned to a menu.

The ShowAll argument lists all Word commands as well as the macros you have created.

If one of the command button names Rename, Delete, or Set is used and followed by another action, multiple MacroEdit commands are recorded.

The NewName$ agrument specifies a new name for the macro; this argument is used with the Rename command button.

1089 Part No. 03915
Kit No. 059-050-007

MacroName$()

~~Syntax:~~ A$ = MacroName$(Count, [Context], [All])

Returns the name of the macro defined in the given context. Count may be in the range of 1 to CountMacros(Context). The name is taken from the list in the given context. MacroName$(0) gives the name of the current macro window, if any. Context is 0 (zero) for global or 1 for document template. If All is True, built-in commands are included.

MacroRecord

~~Syntax:~~ MacroRecord [Name$], [Context], [Description$]

Equivalent to the File Record Macro and the Macro Record dialog boxes. Starts the macro recorder. If Name$ is not given, the next default recording name (Macron) is used. Context is 0 (zero) for global (default) or 1 for document template. The Description$ refers to the text that appears in the status bar if the macro is assigned to a menu.

MacroRun

~~Syntax:~~ MacroRun Name$, [ShowAll]

Equivalent to the Macro Run dialog box. Runs the named macro or command. If ShowAll is 1, built-in commands are included.

MenuMode

~~Syntax:~~ MenuMode

Activates menu mode. Equivalent to pressing Alt or F10.

Mid$()

~~Syntax:~~ Num = 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.

1089 Part No. 03915
Kit No. 059-050-007

MkDir

~~Syntax:~~ MkDir Name$

Creates the directory specified by DirName$.

MoveText

~~Syntax:~~ MoveText

Moves text. Equivalent to pressing F2.

MsgBox

~~Syntax:~~ Message$, [Title$], [Type]

Creates a message box displaying Message$.

Title$ is the title of the message box. If it is not supplied, "Microsoft Word" is the title of the message box. Type determines the symbol and buttons displayed in the box. It is the sum of the values from the following groups:

Type argument	**Displays**
Button:	
0	OK button (default)
1	OK and Cancel buttons
2	Abort, Retry, Ignore buttons
3	Yes, No, Cancel buttons
4	Yes and No buttons
5	Retry and Cancel buttons
Icons:	
0	No icon (default)
16	Hand icon
32	Question icon
48	Exclamation icon
64	Asterisk icon

1089 Part No. 03915
Kit No. 059-050-007

Button action:

0	First button is the default (default)
256	Second button is the default
512	Third button is the default

If Type is negative, then the message is displayed in the status bar and Type must be -1 (display the message permanently), -2 (display until a mouse or key event occurs), or -8 (use the entire status bar width).

MsgBox()

~~Syntax:~~ Num = MsgBox(Message$, [Title$], [Type])

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

Name

~~Syntax:~~ Name OldName$ As NewName$

Renames a file. If the new filename specified already exists, an error is generated.

NextCell

~~Syntax:~~ NextCell

Moves the selection to the beginning of the next cell in a table.

1089 Part No. 03915
Kit No. 059-050-007

NextCell()

~~Syntax:~~ Log = NextCell()

Moves to the next cell. Returns 0 (zero) if there is no next cell.

NextField

~~Syntax:~~ NextField

Moves the selection to the next field result. Skips over marker fields, such as Index Entry fields.

NextField()

~~Syntax:~~ Log = NextField()

Moves to the next field. Returns 0 (zero) if there is no next field.

NextObject

~~Syntax:~~ NextObject

Selects the next object in page view.

NextObject()

~~Syntax:~~ Log = NextObject()

Moves to the next positioned object. Returns 0 (zero) if there is no next object.

NextPage

~~Syntax:~~ NextPage

Moves the insertion point to the beginning of the next page in page view.

1089 Part No. 03915
Kit No. 059-050-007

NextPage()

~~Syntax:~~ Log = NextPage()

Moves the insertion point to the beginning of the next page. Returns 0 (zero) if there is no next page.

NextTab()

~~Syntax:~~ Num = NextTab(Pos)

Returns the position of the next tab stop to the right of Pos. Pos is a number given in points. If more than one paragraph is selected and the tabs do not all match, -1 is returned.

NextWindow

~~Syntax:~~ NextWindow

Moves the selection to the next document window.

NormalStyle

~~Syntax:~~ NormalStyle

Formats the selection in Normal paragraph format.

NormalStyle()

~~Syntax:~~ Num = NormalStyle()

Returns 1 if all of the selection has the Normal style, 0 (zero) if none of the selection has the Normal style, and -1 if part of the selection has the Normal style.

OK

~~Syntax:~~ OK

Terminates a copy or move operation and performs its action.

1089 Part No. 03915
Kit No. 059-050-007

On Error

~~Syntax:~~ On Error Goto Label

~~Syntax:~~ On Error Resume Next

~~Syntax:~~ On Error Goto 0

The On Error control structure allows the programmer to "trap" an error so that the program can perform its own error handling. For more information on On Error, see Macros: Introduction.

OnTime

~~Syntax:~~ OnTime When$, Name$, [Tolerance]

Executes the macro specified by Name$ at the time specified by When$. When$ is a text representation of the time for execution in a 24-hour format. When$ can also include a date string that precedes the time string. If the date is not specified, the macro is run at the first occurence of the specified time. The macro is executed the next time Word is idle after the specified When$. Word does not run the macro if more than Tolerance seconds have elapsed since When$, and the macro has not yet run. If Tolerance is 0 (zero), or not supplied, Word will always run the macro, regardless of how long it is before Word is idle and can run the macro.

Open

~~Syntax:~~ Open Name$ For Mode$ As [#]StreamNumber

Opens the file or device specified by Name$. The Name$ can be a device such as Com1 or Lpt1, and must be enclosed in quotation marks. Do not include the colon following the device name.

OpenUpPara

~~Syntax:~~ OpenUpPara

Adds one line of space before the current paragraph.

1089 Part No. 03915
Kit No. 059-050-007

OtherPane

~~Syntax:~~ OtherPane

Moves the selection to the other pane of the current window.

OutlineCollapse

~~Syntax:~~ OutlineCollapse

Collapses the lowest level of subtext levels under the selected heading.

OutlineDemote

~~Syntax:~~ OutlineDemote

Increases the heading level of the selection by one.

OutlineExpand

~~Syntax:~~ OutlineExpand

Expands the lowest level of subtext under the selected heading.

OutlineLevel()

~~Syntax:~~ Num = OutlineLevel()

Returns the heading level of the specified paragraph. Returns 0 (zero) if the specified paragraph doesn't have a defined level (body text, for example).

OutlineMoveDown

~~Syntax:~~ OutlineMoveDown

Moves the selection below the next visible heading.

1089 Part No. 03915
Kit No. 059-050-007

OutlineMoveUp

~~Syntax:~~ OutlineMoveUp

Moves the selection above the next visible heading.

OutlinePromote

~~Syntax:~~ OutlinePromote

Decreases the heading level of the selection by one.

OutlineShowFirstLine

~~Syntax:~~ OutlineShowFirstLine [On]

If On is omitted, toggles the state. Changes the view of non-heading level text. If On is nonzero, only first line of text is shown, if On is 0 (zero), all text is shown.

Overtype

~~Syntax:~~ Overtype [On]

Without the argument, toggles overtyping mode. If On is nonzero, overtype mode is activated and OVR is displayed in the status bar. If On is 0 (zero), overtype mode is deactivated.

Overtype()

~~Syntax:~~ Log = Overtype()

Returns -1 if overtype mode is on, 0 (zero) if overtype mode is off.

PageDown

~~Syntax:~~ PageDown [Repeat], [Select]

Moves the selection down by Repeat screens. If the Repeat argument is omitted, 1 is assumed. If Select is nonzero, the selection is extended down by Repeat screens. Equivalent to the PgDn key. If Select is 0 (zero) or omitted, the selection is not extended.

1089 Part No. 03915
Kit No. 059-050-007

PageDown()

~~Syntax:~~ Log = PageDown([Repeat], [Select])

Moves the selection down by Repeat pages. Returns -1 if operation was successful, returns 0 (zero) if not.

PageUp

~~Syntax:~~ PageUp [Repeat], [Select]

Moves the selection up by Repeat screens. If the Repeat argument is omitted, 1 is assumed. If Select is nonzero, the selection is extended up by Repeat screens. Equivalent to the PgUp key. If Select is 0 (zero) or omitted, the selection is not extended.

PageUp()

~~Syntax:~~ Log = PageUp ([Repeat], [Select])

Moves the selection up by Repeat pages. Returns -1 if operation was successful, returns 0 (zero) if not.

ParaDown

~~Syntax:~~ ParaDown [Repeat], [Select]

Moves the selection down by Repeat paragraphs. If Repeat is omitted, 1 is assumed. If Select is nonzero, the selection is extended down by Repeat paragraphs.

ParaDown()

~~Syntax:~~ Log = ParaDown([Repeat], [Select])

Moves the selection down by Repeat paragraphs. Returns 0 (zero) if the action cannot be performed. For example, the function returns 0 if the insertion point is at the end of the document.

1089 Part No. 03915
Kit No. 059-050-007

ParaUp

~~Syntax:~~ ParaUp [Repeat], [Select]

Moves the selection up by Repeat paragraphs. If Repeat is omitted, 1 is assumed. If Select is nonzero, the selection is extended up by Repeat paragraphs.

ParaUp()

~~Syntax:~~ Log = ParaUp([Repeat], [Select])

Moves the selection up by Repeat paragraphs. Returns 0 (zero) if the action cannot be performed. For example, the function returns 0 if the insertion point is at the beginning of the document.

PauseRecorder

~~Syntax:~~ PauseRecorder

Stops macro recording until PauseRecorder is executed again.

PrevCell

~~Syntax:~~ PrevCell

Moves the selection to the previous cell.

PrevCell()

~~Syntax:~~ Log = PrevCell()

Moves selection to the previous cell. Returns 0 (zero) when the selection is in the first cell.

PrevField

~~Syntax:~~ PrevField

Moves the selection to the previous field.

1089 Part No. 03915
Kit No. 059-050-007

PrevField()

~~Syntax:~~ Log = PrevField()

Moves selection to the previous field. Returns 0 (zero) when the selection is in the first field.

PrevObject

~~Syntax:~~ PrevObject

Selects the previous object in page view.

PrevObject()

~~Syntax:~~ Log = PrevObject()

Selects the previous object. Returns 0 (zero) when the selection is at the first object or text area.

PrevPage

~~Syntax:~~ PrevPage

In page view, moves the insertion point to the beginning of the previous actual page.

PrevPage()

~~Syntax:~~ Log = PrevPage()

Moves to the previous page. Returns 0 (zero) when the selection is at the first actual page.

PrevTab()

~~Syntax:~~ Log = PrevTab(Pos)

Returns the position of the next tab to the left of Pos. Pos is a number given in points. If more than one paragraph is selected and the tabs do not all match, -1 is returned.

1089 Part No. 03915
Kit No. 059-050-007

PrevWindow

~~Syntax:~~ PrevWindow

Activates the previously active window.

Print

~~Syntax:~~ Print [[#]StreamNumber], Expression

Writes Expression to the file specified by StreamNumber. With no StreamNumber specified, output goes to the status bar.

Read

~~Syntax:~~ Read [#]StreamNumber, Variable(s)

Similar to the Input statement, but removes quotation marks for strings. This statement is used with the Write statement.

RecordNextCommand

~~Syntax:~~ RecordNextCommand

Records the next command at the insertion point in the current macro window.

Rem

~~Syntax:~~ Rem Remarks

~~Syntax:~~ 'Remarks

Inserts explanatory text into the macro. You can use an apostrophe (') instead of a Rem statement. If a Rem statement follows other statements on a line, it must be separated from those statements by a colon (:). A colon is not required before a remark introduced by an apostrophe.

1089 Part No. 03915
Kit No. 059-050-007

RenameMenu

~~Syntax:~~ RenameMenu MenuNumber, NewText$

Renames the top level menu of Menu Number to New Text$. MenuNumber represents the name of a menu. NewText$ replaces the menu name. An ampersand (&) preceding a character makes it the keyboard equivalent to selecting from the menu. For example, "&Programs" becomes Programs when this statement is executed.

The MenuNumber argument values are:

MenuNumber Argument	**Menu**
0	File
1	Edit
2	View
3	Insert
4	Format
5	Utilities
6	Macro
7	Window

Repeat

~~Syntax:~~ Repeat

Repeats the last command.

RepeatSearch

~~Syntax:~~ RepeatSearch

Repeats the most recent search.

ResetChar

~~Syntax:~~ ResetChar

Removes manual character formatting from the selected text. Manual character formatting is

1089 Part No. 03915
Kit No. 059-050-007

formatting that is not applied as a style. For example, you manually format a word or phrase in a paragraph as bold text if the paragraph style is normal text. The text is left with the character formatting of the current style.

ResetChar()

~~Syntax:~~ Num = ResetChar()

Returns 1 if the selected text contains no manual character formatting. Returns 0 (zero) if any manual character formatting is present.

ResetFootnoteContNotice

~~Syntax:~~ ResetFootnoteContNotice

Resets the footnote continuation notice to the default value.

ResetFootnoteContSep

~~Syntax:~~ ResetFootnoteContSep

Resets the footnote continuation separator to the default value.

ResetFootnoteSep

~~Syntax:~~ ResetFootnoteSep

Resets the footnote separator to the default value.

ResetPara

~~Syntax:~~ ResetPara

Removes manual paragraph formatting from the selected text. The text is left with the paragraph formatting of the current style.

1089 Part No. 03915
Kit No. 059-050-007

ResetPara()

~~Syntax:~~ Num = ResetPara()

Returns 1 if the selected text contains no manual paragraph formatting. Returns 0 (zero) if any manual paragraph formatting is present.

Right$()

~~Syntax:~~ A$ = Right$(Source$, Count)

Returns the rightmost Count characters of Source$.

RightPara

~~Syntax:~~ RightPara

Right aligns the selected paragraphs.

RightPara()

~~Syntax:~~ Num = RightPara()

Returns 0 (zero) if none of the selected paragraphs are right aligned, 1 if all of the selected paragraphs are right aligned, or -1 if more than one kind of paragraph alignment is used.

RmDir

~~Syntax:~~ RmDir Name$

Removes the specified directory or subdirectory. Files must first be removed from the subdirectory for this statement to work.

Rnd()

~~Syntax:~~ Num = 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.

1089 Part No. 03915
Kit No. 059-050-007

RulerMode

~~Syntax:~~ RulerMode

Switches to ruler mode.

SaveTemplate

~~Syntax:~~ SaveTemplate

Saves the document template.

Seek

~~Syntax:~~ Seek [#]StreamNumber, Count

Positions file pointer at character Count in the file attached to stream StreamNumber.

Seek()

~~Syntax:~~ Num = Seek([#]StreamNumber)

Returns the current file pointer for the specified StreamNumber.

Select Case

~~Syntax:~~

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

1089 Part No. 03915
Kit No. 059-050-007

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.

For more information on Select Case, see Macros: Introduction.

Selection$()

~~Syntax:~~ A$ = Selection$()

Returns the plain, unformatted text of the selection. The maximum limit on the selection is 32,000 characters or until memory runs out. If the selection is too large, Selection$() is filled with as much of the selection as will fit, and an error is generated. If the selection is an insertion point, the character following the insertion point is returned.

SelectTable

~~Syntax:~~ SelectTable

Selects the table containing the insertion point.

SelType

~~Syntax:~~ SelType Type

Changes the selection highlighting to Type. Type refers to one of the following:

Type argument	Type
0	Hidden
1	Insertion point
2	Selection
4	Dotted selection or insertion point (whatever is current)
5	Dotted insertion point
6	Dotted selection

1089 Part No. 03915
Kit No. 059-050-007

SelType()

~~Syntax:~~ Num = SelType()

Returns the type of the selection highlighting.

SendKeys

~~Syntax:~~ SendKeys Keys$, [Wait]

Sends the keys specified to the active application, just as if they were typed at the keyboard. If Word is not the active application and Wait is -1, Word waits for all keys to be processed before proceeding.

Keys$ is represented by one or more characters, such as a for the character a, {Enter} for the Enter key, and {33} for PgUp.

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}
CapsLock	{capslock}
Clear	{clear}
Del	{delete} or {del}
Down	{down}
End	{end}
Enter	{enter}
Esc	{escape} or {esc}
Help	{help}
Home	{home}
Ins	{insert}
Left	{left}
NumLock	{numlock}
PgDn	{pgdn}
PgUp	{pgup}

1089 Part No. 03915
Kit No. 059-050-007

PrtSc	{prtsc}
Right	{right}
Tab	{tab}
Up	{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, described below.

For example, %{enter} sends the code for Alt+Enter. The code +(eb) specifies EB.

To repeat a key sequence, use the syntax {key number}. For example, {pgdn 20} means press the PgDn key 20 times. Remember to put a space between the key and the number.

SetDirty

~~Syntax:~~ SetDirty [Dirty]

Makes Word recognize the current document as "dirty," or a changed document. If Dirty is omitted or 1, the document is made dirty. If 0 (zero), it makes the document not dirty.

1089 Part No. 03915
Kit No. 059-050-007

SetEndOfBookmark

~~Syntax:~~ SetEndOfBookmark Bookmark1$, [Bookmark2$]

Sets Bookmark2$ to the end point of Bookmark1$. If Bookmark2$ is not supplied, Bookmark1$ is set to its own end.

SetGlossary

~~Syntax:~~ SetGlossary Name$, Text$, [Context]

Defines a glossary entry called Name$ containing the text Text$. Context is 0 (zero) for global, 1 for document template.

SetProfileString

~~Syntax:~~ SetProfileString [App$], Key$, Value$

Sets a value in the current WIN.INI.

App$ is the name of the Microsoft Windows application. If the application is not specified, the string Microsoft Word is used.

SetStartOfBookmark

~~Syntax:~~ SetStartOfBookmark Bookmark1$, [Bookmark2$]

Sets Bookmark2$ to the starting point of Bookmark1$. If Bookmark2$ is not given, Bookmark1$ is set to its own start.

Sgn()

~~Syntax:~~ Num = Sgn(n)

Returns the sign of n. Returns 1 for a positive number, -1 for a negative number, or 0 for zero.

1089 Part No. 03915
Kit No. 059-050-007

Shell

~~Syntax:~~ Shell App$, [WindowStyle]

Starts another program under Microsoft Windows. App$ uses the same format as the File Run command in the Windows MS-DOS Executive, including any switches or arguments that the program accepts. If App$ is the name of a file with an extension specific to an installed application (.DOC for a Word document, for example), the statement starts the application and loads that file.

WindowStyle	**Window type**
0	Minimized window
1	Normal window
2	Minimized window (for Microsoft Excel compatibility)
3	Maximized window
4	Deactivated window

ShowAll

~~Syntax:~~ ShowAll [On]

Without the argument, toggles ShowAll option of the View Preference command. If On is nonzero, shows all invisible objects such as hidden text, tabs, spaces, paragraph marks, and so on. If On is 0 (zero), turns off ShowAll option.

ShowAllHeadings

~~Syntax:~~ ShowAllHeadings

Shows all text in outline view.

ShowHeading1

~~Syntax:~~ ShowHeading1

Shows up to Level 1 headings and hides subordinate headings.

1089 Part No. 03915
Kit No. 059-050-007

ShowHeading2

~~Syntax:~~ ShowHeading2

Shows up to Level 2 headings and hides subordinate headings.

ShowHeading3

~~Syntax:~~ ShowHeading3

Shows up to Level 3 headings and hides subordinate headings.

ShowHeading4

~~Syntax:~~ ShowHeading4

Shows up to Level 4 headings and hides subordinate headings.

ShowHeading5

~~Syntax:~~ ShowHeading5

Shows up to Level 5 headings and hides subordinate headings.

ShowHeading6

~~Syntax:~~ ShowHeading6

Shows up to Level 6 headings and hides subordinate headings.

ShowHeading7

~~Syntax:~~ ShowHeading7

Shows up to Level 7 headings and hides subordinate headings.

ShowHeading8

~~Syntax:~~ ShowHeading8

1089 Part No. 03915
Kit No. 059-050-007

Shows up to Level 8 headings and hides subordinate headings.

ShowHeading9

~~Syntax:~~ ShowHeading9

Shows up to Level 9 headings and hides subordinate headings.

ShowVars

~~Syntax:~~ ShowVars

Displays the list of variables (and their values) currently in use. This statement is useful for debugging macros.

ShrinkFont

~~Syntax:~~ ShrinkFont

Decreases the size of the selected font. Can be used either on the selection or at the insertion point.

ShrinkSelection

~~Syntax:~~ ShrinkSelection

Shrinks the selection to the next smallest unit (word, sentence, paragraph, etc.).

SmallCaps

~~Syntax:~~ SmallCaps [On]

Without the argument, toggles small caps for the entire selection. If On is nonzero, makes the entire selection small caps. If On is 0 (zero), removes small caps from the entire selection.

1089 Part No. 03915
Kit No. 059-050-007

SmallCaps()

~~Syntax:~~ Num = SmallCaps()

Returns 0 (zero) if none of the selection is small caps, 1 if all of the selection is small caps, or -1 if part of the selection is small caps.

SpacePara1

~~Syntax:~~ SpacePara1

Formats the selected paragraphs with single spacing.

SpacePara1()

~~Syntax:~~ Num = SpacePara1()

Returns 0 (zero) if none of the selected paragraphs are single-spaced, 1 if all of the selected paragraphs are single-spaced, or -1 if more than one kind of paragraph spacing is used.

SpacePara2

~~Syntax:~~ SpacePara2

Formats the selected paragraphs with double spacing.

SpacePara2()

~~Syntax:~~ Num = SpacePara2()

Returns 0 (zero) if none of the selected paragraphs are double-spaced, 1 if all of the selected paragraphs are double-spaced, or -1 if more than one kind of paragraph spacing is used.

SpacePara15

~~Syntax:~~ SpacePara15

Formats the selected paragraphs with one-and-one-half line spacing.

1089 Part No. 03915
Kit No. 059-050-007

SpacePara15()

~~Syntax:~~ Num = SpacePara15()

Returns 0 (zero) if none of the selected paragraphs are one-and-one-half spaced, 1 if all of the selected paragraphs are one-and-one-half spaced, or -1 if more than one kind of paragraph spacing is used.

Spike

~~Syntax:~~ Spike

Deletes the selection after copying it to the special glossary called the Spike.

StartOfColumn

~~Syntax:~~ StartOfColumn [Select]

Moves insertion point to topmost position in the currently selected table column. If Select is nonzero, extends the selection.

StartOfDocument

~~Syntax:~~ StartOfDocument [Select]

Moves the selection to the beginning of the document. If Select is nonzero, extends the selection.

StartOfLine

~~Syntax:~~ StartOfLine [Select]

Moves the selection to the beginning of the line. If Select is nonzero, extends the selection.

StartOfRow

~~Syntax:~~ StartOfRow [Select]

Moves insertion point to the leftmost position in the currently selected table row. If Select is

1089 Part No. 03915
Kit No. 059-050-007

nonzero, extends the selection.

StartOfWindow

~~Syntax:~~ StartOfWindow [Select]

Moves the insertion point to the top left corner of the window. If Select is nonzero, extends the selection.

Stop

~~Syntax:~~

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

Str$()

~~Syntax:~~ A$ = Str$(n)

Returns the string representation of value n. Positive numbers have a leading space character.

String$()

~~Syntax:~~ A$ = String$(Count, Source$)

Returns the first character in Source$ repeated Count times. Replacing Source$ with the number m representing the ASCII value of Source$ returns the character with ANSI code m repeated Count times.

StyleName$()

~~Syntax:~~ A$ = StyleName$([Count], [Context], [All])

Returns the name of the style defined in the given context (global or document template). Count may be in the range from 1 to 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 is 0 for global, 1 for document template.

1089 Part No. 03915
Kit No. 059-050-007

Sub...End Sub

~~Syntax:~~

Statement(s)

End Sub

Defines a subroutine. For more information on subroutines, see Macros: Introduction.

SubScript

~~Syntax:~~ SubScript [On]

Without the argument, toggles subscript for the entire selection. If On is nonzero, makes the entire selection subscript. If On is 0 (zero), removes subscript from the entire selection.

SubScript()

~~Syntax:~~ Num = SubScript()

Returns 0 (zero) if none of the selection is subscript, 1 if all of the selection is subscript, or -1 if part of the selection is subscript or superscript.

SuperScript

~~Syntax:~~ SuperScript [On]

Without the argument, toggles superscript for the entire selection. If On is nonzero, makes the entire selection superscript. If On is 0 (zero), removes superscript from the entire selection.

SuperScript()

~~Syntax:~~ Num = SuperScript()

Returns 0 (zero) if none of the selection is superscript, 1 if all of the selection is superscript, or -1 if part of the selection is superscript or subscript.

1089 Part No. 03915
Kit No. 059-050-007

TabLeader$()

~~Syntax:~~ A$ = TabLeader$(Pos)

Returns the leader character of the tab at Pos points. If more than one paragraph is selected and all the tabs don't match, an empty string is returned.

The leader characters returned are blank space, period, hyphen, and underscore.

TabType()

~~Syntax:~~ Num = TabType(Pos)

Returns the type of tab at the given position Pos. If more than one paragraph is selected and all the tabs don't match, -1 is returned. If the tabs match, the type is returned as follows:

If function returns	**Tab type is**
0	Left-aligned
1	Centered
2	Right-aligned
3	Decimal

Time$()

~~Syntax:~~ A$ = Time$()

Returns the current time in the default format.

ToggleFieldDisplay

~~Syntax:~~ ToggleFieldDisplay

Toggles the display between field codes and field results.

UCase$()

~~Syntax:~~ A$ = UCase$(A$)

Returns A$ converted to uppercase.

1089 Part No. 03915
Kit No. 059-050-007

Underline

~~Syntax:~~ Underline [On]

Without the argument, toggles underlining for the entire selection. If On is nonzero, makes the entire selection underlined. If On is 0 (zero), removes underlining from the entire selection.

Underline()

~~Syntax:~~ Num = Underline()

Returns 0 (zero) if none of the selection is underlined, 1 if all of the selection is underlined, or -1 if part of the selection is underlined or more than one kind of underlining is used.

UnHang

~~Syntax:~~ UnHang

Reduces the amount of indent in a hanging indent.

UnIndent

~~Syntax:~~ UnIndent

Removes the indent from the selected paragraphs. The first paragraph is aligned with the previous tab stop.

UnLinkFields

~~Syntax:~~ UnLinkFields

Converts the selected fields to plain text and uses the last result.

UnLockFields

~~Syntax:~~ UnLockFields

Unlocks fields in the current selection for updating.

1089 Part No. 03915
Kit No. 059-050-007

UnSpike

~~Syntax:~~ UnSpike

Empties the Spike glossary and inserts all contents into the document at the selection.

UpdateFields

~~Syntax:~~ UpdateFields

Updates the fields in the selection.

UpdateSource

~~Syntax:~~ UpdateSource

Sends changes in linked Word documents back to their source.

UtilCalculate

~~Syntax:~~ UtilCalculate

Equivalent to the Calculate command on the Utilities menu. The selection is evaluated as a mathematical expression. The result of the evaluation is placed on the Clipboard.

UtilCalculate()

~~Syntax:~~ Num = UtilCalculate([Expression$])

Evaluates Expression. With the argument, this function is equivalent to the = field. Values in Expression can be table cell references. For more information on the = field, see the full Technical Reference. Without an expression, performs the same operation as the UtilCalculate statement, but returns the result rather than placing it on the Clipboard.

UtilCompareVersions

~~Syntax:~~ UtilCompareVersions Name$

Equivalent to the Utilities Compare Versions dialog box. Compares the current document

1089 Part No. 03915
Kit No. 059-050-007

with the document specified by Name$.

UtilCustomize

~~Syntax:~~ UtilCustomize [AutoSave], [Units], [Pagination], [SummaryPrompt], [ReplaceSelection], [Name$], [Initials$], [ButtonFieldClicks]

Equivalent to the Utilities Customize dialog box. Some arguments take measurements in points or numbers. Other arguments correspond to check boxes.

UtilGetSpelling

~~Syntax:~~ UtilGetSpelling FillArray$(), [Word$], [MainDic$], [SuppDic$]

Fills the string array FillArray$ with all available spellings for a word. If Word$ is supplied, that word is used. If it is not supplied, Word uses the word closest to the insertion point. The spellings for each definition are appended in the order they appear in the spelling checker.

```
Sub MAIN
Dim S$(10)
UtilGetSpelling S$(), "color"
For x = 1 To 10
        MsgBox S$(x)
Next x
End Sub
```

UtilGetSpelling()

~~Syntax:~~ Log = UtilGetSpelling(FillArray$(), [Word$], [MainDic$], [SuppDic$])

Fills the string array FillArray$ with all available spellings of a word. If Word$ is supplied, that word is used. If it is not supplied, Word uses the word closest to the insertion point. The spellings for each definition are appended in the order they appear in the spelling checker. Returns 0 (zero) if the word is spelled correctly.

1089 Part No. 03915
Kit No. 059-050-007

UtilGetSynonyms

~~Syntax:~~ UtilGetSynonyms FillArray$(), [Word$]

Fills the string array FillArray$ with all available synonyms for Word$. If Word$ is not supplied, the word nearest the selection is used.

UtilGetSynonyms()

~~Syntax:~~ Log = UtilGetSynonyms(FillArray$(), [Word$])

Fills the string array FillArray$ with all available synonyms for Word$. If Word$ is not supplied, the word nearest the selection is used. Returns 0 (zero) if there are no synonyms available and returns -1 if one or more synonyms are available.

UtilHyphenate

~~Syntax:~~ UtilHyphenate [HyphenateCaps], [Confirm], [HotZone[$]]

Equivalent to the Utilities Hyphenate dialog box. The arguments correspond to check boxes.

UtilRenumber

~~Syntax:~~ UtilRenumber [NumParas], [Type], [StartAt], [ShowAllLevels], [Format$]

Equivalent to the Utilities Renumber dialog box. The arguments correspond to check boxes.

UtilRepaginateNow

~~Syntax:~~ UtilRepaginateNow

Equivalent to the Repaginate Now command on the Utilities menu. Forces repagination of the entire document.

UtilRevisionMarks

~~Syntax:~~ UtilRevisionMarks [MarkRevisions], [RevisionBars], [NewText]

Equivalent to the Utilities Revision Marks dialog box. The arguments correspond to check

1089 Part No. 03915
Kit No. 059-050-007

boxes.

The Search (for next text with revision marking) Accept Revisions or Undo Revisions command button name can be appended.

UtilSort

~~Syntax:~~ UtilSort [Order], [Type], [Separator], [FieldNum[$]], [SortColumn], [CaseSensitive]

Equivalent to the Utilities Sort dialog box.

UtilSpelling

~~Syntax:~~ UtilSpelling [Word$], [MainDic$], [SuppDic$], [IgnoreCaps], [AlwaysSuggest]

Equivalent to the Utilities Spelling dialog box. The arguments correspond to check boxes.

The Delete command button name can be appended to remove the word from the current supplemental dictionary.

UtilSpellSelection

~~Syntax:~~ UtilSpellSelection

Checks the selection. If the selection is only part of a word, the selection is expanded to include the whole word. The default supplemental dictionary is used.

UtilThesaurus

~~Syntax:~~ UtilThesaurus

Lists alternative words for the selection. Equivalent to the Thesaurus command on the Utilities menu.

1089 Part No. 03915
Kit No. 059-050-007

Val()

~~Syntax:~~ Num = Val(A$)

Returns the numeric value of A$.

ViewAnnotations

~~Syntax:~~ ViewAnnotations [On]

Turns on the annotations pane if On is nonzero, turns off the annotations pane is On is 0 (zero). Without the argument, toggles the annotations pane on and off.

ViewAnnotations()

~~Syntax:~~ Log = ViewAnnotations()

Returns -1 if annotations view mode is on, 0 (zero) if annotations view mode is off.

ViewDraft

~~Syntax:~~ ViewDraft [On]

Turns on draft view mode if On is nonzero, turns off draft view mode if On is 0 (zero). Without the argument, toggles draft view mode. If no window is open, the first window opened is opened in draft view.

ViewDraft()

~~Syntax:~~ Log = ViewDraft()

Returns -1 if draft view mode is on, 0 (zero) if draft view mode is off.

ViewFieldCodes

~~Syntax:~~ ViewFieldCodes [On]

Turns on field codes view mode if On is nonzero, turns off field codes view mode if On is 0 (zero). Without the argument, toggles field codes view mode. If no window is open, the first

1089 Part No. 03915
Kit No. 059-050-007

window opened shows field codes.

ViewFieldCodes()

~~Syntax:~~ Log = ViewFieldCodes()

Returns -1 if field codes view mode is on, 0 (zero) if field codes view mode is off.

ViewFootnotes

~~Syntax:~~ ViewFootnotes [On]

Turns on footnotes view mode if On is nonzero, turns off footnotes view mode if On is 0 (zero). Without the argument, toggles footnotes view mode. If no window is open, the first window opened is opened in footnotes view.

ViewFootnotes()

~~Syntax:~~ Log = ViewFootnotes()

Returns -1 if footnotes view mode is on, 0 (zero) if footnotes view mode is off.

ViewFullMenus

~~Syntax:~~ ViewFullMenus

Turns on full menus.

ViewMenus()

~~Syntax:~~ Num = ViewMenus()

Returns the menu state as follows:

Return value	**Menu state**
0	Normal short menus
1	Normal full menus
2	No document short menus

1089 Part No. 03915
Kit No. 059-050-007

ViewOutline

~~Syntax:~~ ViewOutline [On]

Turns on outline view mode if On is nonzero, turns off outline view mode if On is 0 (zero). Without the argument, toggles outline view mode. If no window is open, the first window opened is opened in outline view.

ViewOutline()

~~Syntax:~~ Log = ViewOutline()

Returns -1 if outline view mode is on, 0 (zero) if outline view mode is off.

ViewPage

~~Syntax:~~ ViewPage [On]

Turns on page view mode if On is nonzero, turns off page view mode if On is 0 (zero). Without the argument, toggles page view mode. If no window is open, the first window opened is opened in page view.

ViewPage()

~~Syntax:~~ Log = ViewPage()

Returns -1 if page view mode is on, 0 (zero) if page view mode is off.

ViewPreferences

~~Syntax:~~ ViewPreferences [Tabs], [Spaces], [Paras], [Hyphens], [Hidden], [ShowAll], [DisplayAsPrinted], [Pictures], [TextBoundaries], [HScroll], [VScroll], [TableGridlines], [StyleAreaWidth[$]]

Equivalent to the View Preferences dialog box.

1089 Part No. 03915
Kit No. 059-050-007

ViewRibbon

~~Syntax:~~ ViewRibbon [On]

Turns on the ribbon if On is nonzero, turns off the ribbon if On is 0 (zero). Without the argument, toggles the ribbon. If no window is open, the first window opened is opened with the ribbon.

ViewRibbon()

~~Syntax:~~ Log = ViewRibbon()

Returns -1 if the ribbon is on, 0 (zero) if the ribbon is off.

ViewRuler

~~Syntax:~~ ViewRuler [On]

Turns on the ruler if On is nonzero, turns off the ruler if On is 0 (zero). Without the argument, toggles the ruler. If no window is open, the first window opened is opened with the ruler.

ViewRuler()

~~Syntax:~~ Log = ViewRuler()

Returns -1 if the ruler is on, 0 (zero) if the ruler is off.

ViewShortMenus

~~Syntax:~~ ViewShortMenus [On]

Turns on short menus if On is nonzero, turns off short menus if On is 0 (zero). Without the argument, toggles short menus. If no window is open, the first window opened is opened in short menus.

1089 Part No. 03915
Kit No. 059-050-007

ViewStatusBar

~~Syntax:~~ ViewStatusBar [On]

Turns on the status bar if On is nonzero, turns off the status bar if On is 0 (zero). Without the argument, toggles the status bar.

ViewStatusBar()

~~Syntax:~~ Log = ViewStatusBar()

Returns -1 if the status bar is on, 0 (zero) if the status bar is off.

VLine

~~Syntax:~~ VLine [Count]

Scrolls down vertically by Count lines. If Count is not specified, one line is the default. A negative Count scrolls up.

VPage

~~Syntax:~~ VPage [Count]

Scrolls down vertically by Count screens. If Count is not specified, one screen is the default. A negative Count scrolls up.

VScroll

~~Syntax:~~ VScroll Percentage

Scrolls vertically the specified percentage of the document length.

VScroll()

~~Syntax:~~ Num = VScroll()

Returns the current vertical scroll position as a percentage of the document's size.

1089 Part No. 03915
Kit No. 059-050-007

While...Wend

~~Syntax:~~

```
        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.

Window()

~~Syntax:~~ Num = Window()

Returns the number of the currently selected window. The number ranges from 1 to the number of open windows. The number corresponds to the number on the Window menu.

Window1

~~Syntax:~~ Window1

Selects Window 1. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window2

~~Syntax:~~ Window2

Selects Window 2. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window3

~~Syntax:~~ Window3

Selects Window 3. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

1089 Part No. 03915
Kit No. 059-050-007

Window4

~~Syntax:~~ Window4

Selects Window 4. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window5

~~Syntax:~~ Window5

Selects Window 5. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window6

~~Syntax:~~ Window6

Selects Window 6. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window7

~~Syntax:~~ Window7

Selects Window 7. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window8

~~Syntax:~~ Window8

Selects Window 8. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

Window9

~~Syntax:~~ Window9

1089 Part No. 03915
Kit No. 059-050-007

Selects Window 9. This number corresponds to the number on the Window menu. If you select a nonexistent window, an error is generated.

WindowArrangeAll

~~Syntax:~~ WindowArrangeAll

Arranges all open windows so that windows do not overlap.

WindowName$()

~~Syntax:~~ 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 current window is returned.

WindowNewWindow

~~Syntax:~~ WindowNewWindow

Equivalent to the New Window command on the Window menu. Creates a copy of the current window.

WindowPane()

~~Syntax:~~ Num = WindowPane()

If the window isn't split or if the top pane of the current window is selected, returns 1. If the bottom pane is selected, returns 3.

WordLeft

~~Syntax:~~ WordLeft [Repeat], [Select]

Moves the insertion point left by Repeat words, extending the selection if Select is nonzero.

1089 Part No. 03915
Kit No. 059-050-007

WordLeft()

~~Syntax:~~ Log = WordLeft([Repeat], [Select])

Moves the selection left by Repeat words. Returns 0 (zero) if the action cannot be performed. For example, the function returns 0 if the insertion point is at the beginning of the document.

WordRight

~~Syntax:~~ WordRight [Repeat], [Select]

Moves the insertion point right by Repeat words, selecting if Select is nonzero.

WordRight()

~~Syntax:~~ Log = WordRight([Repeat], [Select])

Moves the selection right by Repeat words. Returns 0 (zero) if the action cannot be performed.

WordUnderline

~~Syntax:~~ WordUnderline [On]

Without the argument, toggles word-only underlining for the entire selection. If On is nonzero, makes the entire selection word-only underlining. If On is 0 (zero), removes word-only underlining from the entire selection.

WordUnderline()

~~Syntax:~~ Log = WordUnderline()

Returns 0 (zero) if none of the selection is word underlined; 1 if all of the selection is word underlined; or -1 if part of the selection is word underlined or more than one kind of underlining is used.

1089 Part No. 03915
Kit No. 059-050-007

Write

~~Syntax:~~ Write [#]StreamNumber, Expressions

Writes the arguments to StreamNumber including delimiters so they can be read by the Read 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

1089 Part No. 03915
Kit No. 059-050-007

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.

1089 Part No. 03915
Kit No. 059-050-007

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.

1089 Part No. 03915
Kit No. 059-050-007