

 **MACRO TOOLS for Word for Windows 6.0x and 7.0**





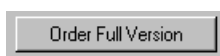
Tools to facilitate creating Word for Windows 6.0x and 7.0 WordBasic macros.

CONTENTS

Double-click page number below to jump to a topic.

Double-click the button at the end of each section to return.

Use Outline view for quick navigation.

	TOOLSFUNCTIONSANDSUBROUTINES
	TOOLSCommandHELPER
	TOOLSDOCUMENTVARIABLES
	VERSION HISTORY
	AGREEMENT & ORDER INFORMATION

Note: The Demo Version of Macro Tools has restricted functionality as described below. The full versions do not have these limitations.

 **Tools Functions And Subroutines****MANAGING COMPLEX MACROS AND APPLICATIONS**

Ever have trouble finding a function or subroutine in a large macro? You couldn't quite remember the name or the direction you needed to scroll relative to the insertion point? Then had trouble getting back?

Ever wonder whether the name of the function or subroutine you are calling in a library macro is LibraryName.FunctionOrSubName or LibraryName.SubOrFunctionName? Have you even *once* been able to remember its parameters?

Ever manage to remember everything but make a 1-character typing error that took a half-hour to find and correct?

If any of this sounds familiar, you are creating ambitious custom applications using WordBasic, and you need this macro!

TOOLSFUNCTIONSANDSUBROUTINES collects names of a macro's functions and subroutines in a list box. Further, if the function or subroutine you want is in a different macro, you can choose the macro whose functions and subroutines

TOOLSFUNCTIONSANDSUBROUTINES will list. From *any* macro in *any* open template. Even if the macro window isn't open!

Once you choose a function or subroutine, you can go directly to it, opening or activating the other window if needed. Or, you can insert the name of the function or

subroutine and parameters in the active macro. Here, too, closed or inactive macro windows are no barrier to TOOLSFUNCTIONSANDSUBROUTINES.

Previews show the selected macro's description and the first five lines of functions and subroutines.

INSTALLATION

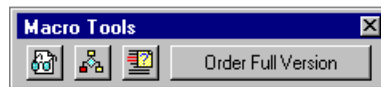
Demo Version: The demo version can not be used as a global macro, therefore there is no point in installing it. This template must be open and active for the macro to operate.

Full Version: The easiest way to install TOOLSFUNCTIONSANDSUBROUTINES is to copy MCROTOOL.DOT to Word's "Startup" directory. Then restart Word.

I've included a Functions And Subroutines tool on the Macro Tools toolbar. I recommend copying this tool to the macro toolbar for your Normal template, next to the "Macro" tool. This way, it is always available when you are editing macros. To copy a tool, open both toolbars, hold down Alt and Ctrl and drag the tool to the new toolbar. To start the macro, a macro window must be active, so there is little point in installing the macro on a menu if you always leave the Macro toolbar open while editing macros.

If you use the Macro Tools toolbar and copy the macro to another template using Organizer, remember to copy the toolbar with it.


USING ToolsFunctionsAndSubroutines



1. If the Macro Tools toolbar is not visible, display it now. (View/Toolbars/Macro Tools)
2. I have included a sample macro for demonstration purposes. Open FunctionAndSubSampleMacro (Tools/Macro/FunctionAndSubSampleMacro/Edit). Read through the instructions here, first. A brief summary is included in the header of FunctionAndSubSampleMacro. The active window must be a macro to start TOOLSFUNCTIONSANDSUBROUTINES, so you may not be able to see these instructions.

Demo Version: FunctionAndSubSampleMacro *must be the active window* to start TOOLSFUNCTIONSANDSUBROUTINES. However, it can display, Go To and Insert functions and subroutines from any open template, just like the Full Version.

Full Version: Any macro can be active when you start TOOLSFUNCTIONSANDSUBROUTINES. The active macro need not even be the macro whose function or subroutine for which you are looking!

1. *Activate the macro window first*, then click the Functions and Subroutines  tool. The Functions and Subroutines dialog box appears.

2. The Functions And Subroutines Available In list lets you select from all macros in all open templates, and all macros in the same templates as any open macros. When you select a macro, the Functions And Subroutines list changes to show you what is in the macro you just selected.
3. Select a function or a subroutine from the list. Choose Go To or Insert.

Go To

Moves the selection to the subroutine or macro you chose. If the subroutine or function is in a different macro, the macro opens or activates as needed.

Insert

Inserts the name of the subroutine or function and its parameters, if any, into the active macro at the insertion point. This lets you insert a call to a macro or subroutine with neither typing nor typos. Insertion syntax is context-sensitive.

C O N T E X T	S Y N T A X
Selected subroutine or function is in the active macro	SubOrFunctionName (Param1, Param2\$, ...)
Selected subroutine or function is <i>not</i> in the active macro	MacroName.SubOrFunctionName (Param1, Param2\$, ...)
“Sub MAIN” is selected to call another macro. MAIN has no parameters.	MacroName
“Sub MAIN” is selected to call another macro. MAIN <i>has</i> parameters.	MacroName.MAIN (Param1, Param2\$, ...)

Note that this last form works, but is as far as I know undocumented. You can call MAIN with parameters from another macro, but it can not be assigned to a speed key, menu command or toolbar, or run from the ToolsMacro dialog box or macro command. It can be very handy, though, because MAIN subroutines have special privileges such as shared variables.

Many are the bugs caused by incorrectly entered subroutine or function calls and missing or misplaced parameters. Using TOOLSFUNCTIONSANDSUBROUTINES ensures you won't waste time on this again.

COMPATIBILITY

TOOLSFUNCTIONSANDSUBROUTINES works with any *available* macro. You may never need these guidelines, but this wouldn't be documentation unless it got into the details. A macro is *available* when—

- It is not execute-only (it is “unencrypted”) and
 - it is open in an editing window, or

- it is contained in a template that is open and is not protected for forms or annotations, or
- it is in the same template as another macro that is open in an editing window,

or

- It is not an execute-only macro, and it is in the NORMAL.DOT template, and NORMAL.DOT is not protected for forms or annotations.

These are a lot of rules, but the “bottom line” is that almost every macro you use when creating applications will work with it! What may not be obvious from the rules is that if a macro is in a template protected for annotations or revisions, TOOLSFUNCTIONSANDSUBROUTINES can still access the macro if it was open before you protected the template. Also, templates protected for revisions work just fine.

WordBasic has no commands like “CountFunction()” and “FunctionName\$(Count)”, so TOOLSFUNCTIONSANDSUBROUTINES has to parse the macro code for Sub and Function statements. This makes it somewhat syntax-sensitive. Fortunately, there are not that many possibilities (I hope). Word helps by reformatting open macros when it saves or runs them and checking syntax when it executes.

The only known condition tolerated by Word macro syntax but not the TOOLSFUNCTIONSANDSUBROUTINES macro is if the Sub or Function command is not at the left margin; in other words, when the Sub or Function keyword is preceded by tabs, spaces or a colon. Making the macro tolerant of this condition would slow it down unacceptably. I have a hard time imagining anyone coding this way on purpose or by mistake, but if you do, the function or subroutine will be missing in the Functions And Subroutines list.

ToolsFunctionsAndSubroutines fully supports up to 500 macros open in unlimited numbers of templates, each up to 65,280 characters and up to 500 (total) functions and subroutines per macro. It offers partial support of macros exceeding these limits: The first 500 functions and subroutines that start within the first 65,280 characters are listed. If there are more than 500 (total) macros among all open templates, the *last* 500 as they would be listed on the Window menu (alphabetical order) are shown. (Yes, the *last* 500. Don’t ask.) A message box alerts you to possible omissions. Any macro larger than 65,280 characters or 500 functions and subroutines comes by the name “Macro” honestly!

PERFORMANCE TIPS

Small macros like FunctionAndSubSampleMacro are read in very quickly, but large macros can take several seconds. TABLE.WIZ:StartWizard, for example, takes a hair under 5 seconds on my 486DX2/50 when it is the only open window. On my Pentium/120, if you blink, you miss it. On most 486 or better machines, it’s still much faster than opening and scrolling through macros manually or typing names into Edit/Find, but even on a high end PC, you may find it lags a little, sometimes.

I spent a lot of time optimizing algorithms, but Word does not provide any *built-in* means (that is, macro commands) for obtaining function or subroutine names. This means the macros themselves have to be opened and analyzed. This soaks up processor cycles. If you want to make TOOLSFUNCTIONSANDSUBROUTINES faster, try these tips:

- Full Version only: If you are editing a large macro and want to Go To a subroutine in a much smaller one, consider switching to that window first, then starting TOOLSFUNCTIONSANDSUBROUTINES. This is because TOOLSFUNCTIONSANDSUBROUTINES loads the subroutine and function names from the active macro automatically as it opens. Of course, if you plan to use the Insert button, the active macro must be the one in which you want the text inserted.
- Leave as few templates and macros open as possible, but...
- ...leave at least one macro from each template open. Having any macro in a template open is *much* faster than a template with no macros open. Leaving the “target” macro open but inactive is slightly faster. The fastest performance is achieved when the “target” macro is the active macro (Full Version only).

There are also a few things you can do when writing macros. Most code jockeys will be unwilling to modify their coding style to suit this macro; nonetheless, these techniques speed up TOOLSFUNCTIONSANDSUBROUTINES by bypassing some of its parsing code.

- Do not split a SUB or FUNCTION line using a backslash.
- Do not end a SUB or FUNCTION line with a comment (').

The macro is compatible with these techniques, but must filter them to display and insert them properly, which requires more processing.

Finally, distributing subroutines and functions among multiple macros in a clear, organized way is as beneficial to the efficiency of TOOLSFUNCTIONSANDSUBROUTINES as it is to your own.



ToolsCommandHelper

WORDBASIC HELP MADE MORE HELPFUL

FileSave is a great WordBasic command. It's short, easy to type, has only a statement form, and takes no parameters. Would that all of WordBasic were as simple and elegant as that FileSave command! But many commands have parameters that go on and on and on...

You probably think it's pretty cool that you can click on a WordBasic command in a macro window and press F1 for context-sensitive help on that command. But that's only half the battle, isn't it. You still have to type the command into your macro.

Word 7 adds a new complication. The Help window resizes each time you call it by pressing F1 or using Help Topics—and completely covers the code you’re working on. Because of this, Always On Top becomes essentially unusable. (What were they thinking?)

Did you ever wish there was a magic button you could click and—

- Insert the command syntax into your macro?
- With arguments?
- Ready to edit?

That would make FilePageSetup or FormatParagraph as fast and painless as File-Save.

TOOLSCOMMANDHELPER is that magic button.

INSTALLATION

Demo Version: The demo version can not be used as a global macro, therefore there is no point in installing it. This template must be open and active for the macro to operate.

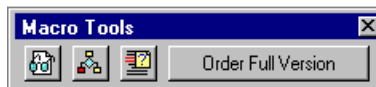
Full Version: The easiest way to install TOOLSCOMMANDHELPER is to copy MCROTOOL.DOT to Word’s “Startup” directory. Then restart Word.

WordBasic Help must be installed. If it is not, run Word (or Office) SETUP and install it.


I’ve included a Command Helper tool on the Macro Tools toolbar. I recommend copying this tool to the macro toolbar for your Normal template. This way, it is always available when you are editing macros. To copy a tool, open both toolbars, hold down Alt and Ctrl and drag the tool to the new toolbar. In order to start the macro, a macro window must be active, so there is little point in installing the macro on a menu if you always leave the Macro toolbar open while editing macros.

If you use the Macro Tools toolbar and copy the macro to another template using Organizer, remember to copy the toolbar with it.

USING ToolsCommandHelper



1. If the Macro Tools toolbar is not visible, display it now. (View/Toolbars/Macro Tools)
2. I have included a sample macro for demonstration purposes. Open CommandHelperSampleMacro (Tools/Macro/CommandHelperSampleMacro/Edit). Read through the instructions here, first. More detailed instructions are included in the header of FunctionAndSubSampleMacro.

3. **Demo Version:** CommandHelperSampleMacro *must be the active window* to start TOOLSCOMMANDHELPER.
4. **Full Version:** Any macro can be active when you start TOOLSCOMMANDHELPER.
5. Continue with the instructions in the macro window. When you need it, the Command Helper tool looks like this: .

NOTES

TOOLSCOMMANDHELPER works with almost every macro command without problems. The few that break it do so in a forgiving and still-useful way. Odds are good you will never encounter any of these situations, but nobody likes surprises, so...

1. TOOLSCOMMANDHELPER is designed for use with WordBasic Help topics for WordBasic macro commands only. It cannot always detect when other types of Help topics are displayed and will act unpredictably in those situations.
2. Microsoft never intended that WordBasic Help be read by anything other than one or more eyes connected directly to a brain. They cannot therefore be faulted for having been almost-but-not-quite totally consistent from Topic to Topic in the way they arranged the text. In fact, it's near-miraculous (not to mention fortunate for TOOLSCOMMANDHELPER) that they used a layout that can be parsed at all by a macro.
3. TOOLSCOMMANDHELPER was tested with every WordBasic command in the WRDBASIC.HLP file accompanying the US-English version of Word for Windows 6.0 and 7.0. For Word 6, both Windows 3.1x and Windows 95 Help engines were tested. There are a few Topics with which TOOLSCOMMANDHELPER has trouble. TOOLSCOMMANDHELPER detects conditions it can not accurately parse and warns you to check what it inserts.

Rather than hard-code special cases into the macro which could be obsoleted by other versions of the Help file, I was able to devise a "fallback" parsing routine that is accurate *nearly all of the time*. When it fails, you will get the macro syntax concatenated with at most one sentence from the command description that follows the syntax.

Two commands, BorderNone and DlgLoadValues, do not offer both forms for you to choose from, and simply inserts the first form. REM inserts both forms on the same line. (Fortunately, REM is the one command where this is harmless! But you would more likely use Microsoft's Add/Remove REM tool or type an apostrophe.)

4. You may notice that the screen "blinks" when you use TOOLSCOMMANDHELPER with certain commands. Block commands (Sub...End Sub, While...Wend, If...Then...Else If, and so on) must briefly maximize, then restore the Help window. On a fast machine with accelerated video, it's ugly but over

quickly. On a '386SX with VGA, it's probably a bit more jarring. If maximize and restore events have sounds assigned to them on your machine, you're gonna hear it no matter how fast your machine is. Then again, you can probably type the block commands in your sleep, so you may never use `TOOLSCOMMANDHELPER` to insert them!

5. If WordBasic Example is open, `TOOLSCOMMANDHELPER` must close it for proper operation.



ToolsDocumentVariables

MANAGING DOCUMENT VARIABLES MADE EASY

WordBasic macros can store settings related to a document in Document Variables. Document Variables are like INI file or Registry entries, but with these advantages:

- They are stored in the document and so use no additional files
- INI files or Registry entries are well suited to storing settings that serve many files, but document variables are saved in the document, and are thus specific to that file.
- With them, macros can record user selections for later use by other macros, or by the same macro re-invoked by the user.

Macros using document variables are harder to debug, however, due to lack of a document variable editor/viewer in Word. There is no easy way to determine if a variable is correct, or to set variables for testing. I wrote the `TOOLSDOCUMENTVARIABLES` macro to fill that void. Finding it indispensable in creating a major custom application for my company that creates “smart documents” that have to track large amounts of document-specific information, I have since “polished it up” for this offering.

With the `TOOLSDOCUMENTVARIABLES` macro, contained in this template, blind faith is no longer required when creating macros that use Document Variables.

WHAT THE MANUAL WON'T TELL YOU

If you are just getting started with document variables, you may find these empirical observations helpful—

- Like INI file settings, document variable names and values can include spaces and extended ANSI characters. Unlike INI strings, they can also include tabs and carriage returns.
- Like INI file settings, document variables are text, so your macro must use the `Str$()` and `Val()` functions to convert to and from numeric values. Also like INI files, the variable name need *not* have a “\$” suffix.

- As *Count* increments in `GetDocumentVarName$(Count)`, document variable names are returned in alphabetical order.
- Document variables in templates are inherited by documents based on them. Once a document is created, however, changes to the document variables in the template do not affect documents attached to them.
- Document variables in global templates are not available to macros.
- Document variables raise the active document's "dirty" flag when set.
- The ability to set, read and delete document variables is unaffected by document protection.

INSTALLATION

Demo Version: The demo version operates only on `MCROTOOL.DOT`, so there is no point in installing it.

Full Version: The easiest way to install `TOOLSDOCUMENTVARIABLES` is to copy `MCROTOOL.DOT` to Word's "Startup" directory.

I've included a Document Variables tool on the Macro Tools toolbar. I recommend copying this tool to the macro toolbar for your Normal template, next to the "Show Variables" tool. To copy a tool, open both toolbars, hold down Alt and Ctrl and drag the tool to the new toolbar. This way, it is always available when you are editing macros.



I have also set the Macro Tools template to add a Document Variables command to your Tools menu whenever you load the template. This helps debug macro problems when the active window is a document instead of a macro.

If you use the MacroTools toolbar and copy the macro to another template (using the Organizer), remember to copy the toolbar with it. Also, set the menu command using Tools/Customize.

USING ToolsDocumentVariables

I have set a few document variables in this template for demonstration purposes. In the demo version, this file must be the active document for `TOOLSDOCUMENTVARIABLES` to work. The full version works with any document or template.



1. If the Macro Tools toolbar  is not visible, display it now. (View/Toolbars/Macro Tools)
2. Click the Document Variables  tool. The Document Variables dialog box appears.

Use of the dialog is straightforward. When you select a variable on the list, its name and value appear in the edit boxes below the list. You can edit these and click Set to change a document variable, or Delete to delete one.

To create a new document variable, type in a new name and value, then click Set.

Tip: The Document Variables dialog is similar to Word's ToolsAdvancedSettings dialog box. This is a good command, if you don't know about it. With it, you can change settings in WIN.INI or other INI files interactively. Run the ToolsAdvanced-Settings Word command if you've never seen it.

VERSION HISTORY

Wherein I am brutally honest about my shortcomings and lapses.

VERSION 1.0: 6 FEBRUARY 1995

Initial release uploaded to CIS.

VERSION 1.01: 16 FEBRUARY 1995

TOOLSFUNCTIONSANDSUBROUTINES

Bugfix for macros larger 65,280 characters. Macro size was not detected correctly.

VERSION 1.1: 11 MARCH 1995

TOOLSFUNCTIONSANDSUBROUTINES

Added macro description and subroutine/function preview to dialog box.

Optimized code that builds macro list. Now nearly 50% faster when multiple templates open.

Cosmetic fix: Macros in templates not open, but attached to open document windows now excluded from the list of available macros.

Bugfix for when active macro window split into panes. Now Goto scrolls the correct pane, and Insert uses the correct insertion point.

TOOLSDOCUMENTVARIABLES

Fix to dialog text truncated when using VGA screen font.

ORDERFULLVERSION

Removed MacroToolsIcon from AutoText, dialog box and order form to make macro less dependent on user's Word environment. Well, it seemed like a good idea at the time...

VERSION 1.11: 13 MARCH 1995

Internal version not publicly released.

VERSION 1.12: 25 MAY 1995

TOOLSFUNCTIONSANDSUBROUTINES

A user reported an error I was unable to duplicate. Nonetheless, there was another efficient route to accomplish what the macro was tripping over so I changed the routine and the problem reportedly went away. Another satisfied customer. Now the same fix is in the "official" version in case anyone else has the same problem and just never told me about it.

Also, an error in the sort procedure sorted functions and subroutines only if there was more than 2. I'll bet no one noticed. But, it now sorts if there is more than one.

VERSION 1.13: 19 JUNE 1995

TOOLSFUNCTIONSANDSUBROUTINES

Fixed condition where macro character number 65,280 occurs in the middle of a "Function" or "Sub" line.

VERSION 1.14: 24 JULY 1995

Minor fixes to the demo version to make it more convenient for the user and prepare it for inclusion in The Cobb Group's shareware offering.

AUTOOPEN

Added a macro that opens the Macro Tools toolbar as the Demo Version of the template opens.

VERSION 1.2: 8 SEPTEMBER 1995

TOOLSFUNCTIONSANDSUBROUTINES

Lengthened macro list drop-down.

Fixed condition where macros in templates attached to RTF files (and potentially other file types) were mistakenly listed.

TOOLSCOMMANDHELPER

Debut of this Macro Tool.

VERSION 1.21: 7 OCTOBER 1995

TOOLSFUNCTIONSANDSUBROUTINES

Made compatible with WinWord 7.0 (retaining 6.0x compatibility).

Fixed condition where macro crashed if a macro was open but its template window was not.

TOOLSCOMMANDHELPER, ORDERFULLVERSION

Made compatible with WinWord 7.0 (retaining 6.0x compatibility).

DISCLAIMER AND AGREEMENT

Users of the Full and Demonstration versions of `TOOLSDOCUMENTVARIABLES`, `TOOLSFUNCTIONSANDSUBROUTINES`, `TOOLSCOMMANDHELPER` and/or the Word template in which they are delivered, `MCROTOOL.DOT` (“the Software”) must accept this disclaimer of warranty:

“The Software is supplied as is. The author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The author assumes no liability for damages, direct or consequential, which may result from the use of the software.”

The Demonstration Version is provided at no charge to the user for evaluation. Feel free to share it with your friends, but please do not give it away altered or as part of another system. You may not distribute the Full Version.

If you find this program useful and would like to obtain the Full Version, you can purchase it as described below, and in the `ORDERFULLVERSION` macro. This will license one copy for use on any one computer at any one time. Think of this as analogous to use of a book. For example, by any number of people Software may use this and may freely move it from one computer to another, so long as there is no possibility of its being used simultaneously on two or more computers. Just as two different persons cannot read a book at the same time.

Contact Jeffrey R. Vandervoort for site license arrangements.


Anyone distributing the Software for any kind of remuneration must first contact Jeffrey R. Vandervoort at the address below for authorization.

You are encouraged to pass a copy of the Demonstration Version of the Software along to your friends for evaluation, provided that you give them the complete `MCROTOOL.DOT` file, including all macros and this text. All purchasers will receive a copy of the latest version of the Software.

ORDERING: FULL VERSION AND UPGRADES

You can order Macro Tools through CompuServe on-line registration and receive Macro Tools through CompuServe e-mail, or you can send an order form by US mail.

- To order the Full Version of Macro Tools on-line, GO SWREG, Registration ID 7509. Price is \$10.00, delivered *via* CompuServe e-mail.

- To order an Update on-line, request it by sending e-mail to the CIS ID below. No charge to registered users of versions 1.0x through 1.2x, delivered *via* CompuServe e-mail.
- To order by mail, click the Order Full Version tool on the Macro Tools toolbar  or run the ORDERFULLVERSION macro in MCROTOOL.DOT. Prices and shipping method options are shown in the dialog box and order form.

Thank you for supporting this product.

COMMENTS, SUGGESTIONS AND QUESTIONS

Please e-mail me—

Jeffrey R. Vandervoort

CompuServe ID: 74651,2175