Contents

Overview

Description:

Macro Mania is a powerful, yet very easy-to-use macro program that works with *any* windows program. Macro Mania will launch programs (or switch to currently running programs) and then send any keystroke to that program with one simple command, just as if you had manually started the program and typed in all the keystrokes yourself. Macro Mania takes the powerful concept of macros to the next level. Because it runs external to other programs, it can launch new or activate already running programs, transfer information between programs, wait for one program to finish and then start another, schedule macros to run unattended, and more.

If you are like many of the *Macro Maniacs* that continue to emerge, you will find yourself creating macros for all kinds of tasks that require repetitive keystrokes. Some of them will be permanent, crucial parts of your daily activities and others will be temporary macros you quickly create to save you a lot of time and keystrokes for one-time projects. Since its introduction back in 1995, Macro Mania continues to receive positive comments from professional software reviewers as well as from *many* Windows users that believe, as we do, a PC should work for us, not vice versa <smile>.

An example of the *many* uses for *Macro Mania*:

I have a contact management program that stores peoples names, addresses, e-mail addresses, etc. Occasionally I want to send an *e-mail newsletter* to certain people in my list. Because I do not want to have to maintain two lists (one in my e-mail program and one in my regular contact management program), and I want to send a message to most, but not all of the people in my list, I need a way to use my contact management program to select the people I want, then have it send e-mail messages to only those people I have selected.

Using *Macro Mania*, I can have it send the keystrokes to my contact management program that filters the list for me and then moves to the first entry. Then it starts my e-mail program, sends the keystrokes I would use to compose a new message, automatically plugs the names and e-mail addresses, switches over and captures the text of a letter I have already typed out in Notepad, switches back to the e-mail program and pastes the contents of the letter, saves the message, then moves on to the next entry to do the same thing. With one click of the mouse I have maneuvered among three different Windows programs, moving information from one program to the other, and have sent dozens of keystrokes! What would normally take an hour (or several hours depending on the number of entries) to do now takes about 1 minute!!!

Of course, this example just explains one of the many uses our customers have come up with, we are sure you will find your own terrific time-savings tasks to assign Macro Mania. **- ENJOY!!!**

Activate

Description:
Brings an already running copy of a program to the front of your Windows environment. (See Starting Programs for more discussion.)

Syntax 1:
ACTIVATE [Title Bar]

Title Bar is the exact caption that is found at the top of the programs windows (like the *Macro Mania Help* title bar shown above).

Syntax 2:
Activate2 [Title Bar Text]

Title Bar Text is any text within a title bar. This is useful where only a partial match is needed or desired.

Example 1:
ACTIVATE NOTEPAD - thefile.txt

Example 2:
ACTIVATE2 Notepad

Notes:

The Activate command, though useful, is generally not needed since the Run command can also pull currently running programs to the front and, if not found, automatically start the program for you. Additionally, the RUN command does not require you to know the Title Bar Caption, just the name of the EXEcutable which is generally more static.

Neither *Activate* nor *Activate2* requires the *Title Bar* to match the case (upper or lower case).

An error occurs when you use the Activate command and the Title Bar is incorrect or the program is not already running.

Activate2 is a powerful extension of the Activate command when the Title Bar changes (as when it also has the name of an open file).

It does not change whether it is maximized or minimized. If you are not sure if the macro might be activating a program that has been minimized, one trick is to send *Alt-space, R* using the SEND command (translated as *SEND* %{ }*R*). This usually causes the program to **R**estore to a *normal* window if it is minimized. (See manipulate the state of windows for more discussion.)

Beep

Description:
Sends a beep to any normal PC Speaker.

Syntax:
Beep

Notes:
Useful for bringing attention to the computer. Particularly useful to put at the end of long macros or immediately before a pause to indicate you may begin typing.

See Also:
PlayWav

ChDir

Description:

Like the DOS command, this changes the working directory to the directory you specify. Issue this command if the program you wish to run using the Run command needs to find support files. (Issuing this command before the RUN command is like specifying the *Working Directory* when you create a program item in the Windows Program Manager.)

Syntax:

CHDIR [path]

path is a string expression that identifies which directory becomes the new default directory. This argument must contain fewer than 128 characters and has the following full syntax:

CHDIR [drive:] [\ ]directory[\directory] . . .

The argument drive is an optional drive specification; the argument directory is a directory name. If you omit drive, CHDIR changes the default directory on the current drive.

Notes:

CHDIR may be abbreviated with *CD*.

End

Description:
Causes the Macro Mania program to completely end.

Syntax:
END

Notes:
This command is especially useful if you wish to run a macro, perhaps from another program, then want Macro Mania to stop running entirely. To simply exit an individual macro, see the Exit command.

Exit

Description:
Causes the macro to exit and quit running.

Syntax:
EXIT

Notes:
This command is especially useful for debugging your macros or dynamically changing where the macro should end without having to comment a lot of lines or delete part of a macro script you may want to use later. For debugging, if you are not sure where your macro is not working correctly, place this command a few lines above where you think the error is happening. Then gradually move it down one line at a time until the line with the error is identified.

To end the Macro Mania program entirely, see the End command.

GetInput

Description:
Gives a prompt within a macro, allowing for input of small amounts of dynamic data that can later be sent with the SendInput command (or Send command if a *variable* is used).

Syntax:
GetInput [{var-?}] [Prompt]

{*var-?*} is optional--however, when present, the contents of whatever is received is put into the specified variable where you can later use any of the Macro Mania commands that support variables with that variable.

Prompt is any text (approximately 255 characters maximum) that you want displayed when the Input Box is shown. (Usually a description of what should be typed in the box.)

Example:
GetInput {var-b} Please enter your first and last name.

Notes:
This command is useful for getting file names, search strings, and other data that may change each time a macro is run.

Once you use the GetInput command, it is necessary to use either the Run or Activate command and later a SendInput command (or Send command if the contents was put in a variable) to send the information that was put into the Input Box.

By putting the input into several different variables, you can issue several GetInput statements in a row without the need to issue a SendInput command to prevent overwriting the previous contents. Thus, using variables with the GetInput command and then using the Send command to send the contents of the variable is much more powerful than using the GetInput and later the simple SendInput command. (For backward compatibility, the SendInput command is still available, but it is really not needed except for very simple cases.)

If no input is put into the box or the cancel button is selected, the macro asks if you wish to abort the macro. Yes causes the macro to stop, no causes the macro to continue without any input. (When the SendInput command is encountered without any text to be sent, it will ask a similar question.)

Detailed Example:

```
GETINPUT Enter the file to open.
RUN notepad.exe
SEND %Fo
SENDINPUT
SEND {enter}
```

Macro

Description:
Runs another macro you have programmed, allowing you to chain macros together.

Syntax:
MACRO [n]

n is a number indicating the number of the macro you wish to run. Refer to the number displayed in the Macro Editor or on the status bar on the Main Screen.

Example:
MACRO 3

Notes:
Macro Mania prevents you from using the MACRO command to call itself, otherwise it would get into a continuous loop. It is possible to call a macro that, in turn, calls the macro that called it, so be sure you refer to the correct macro number and avoid getting yourself into such a continuous loop.

The macro command essentially allows you to create subroutines within your macros. By putting repeat commands in certain macros, you can also create nested loops, etc. too. If you only plan to use a macro as a subroutine from another macro, you can turn its visible property to off from where you select the icons for buttons, preventing a macro from showing up as a button on the main Macro Mania screen.

If your macro does not require a repeating subroutine, you may also use the Goto command to branch within a single macro and perhaps better keep your macro intact and easier to manage.

Minimize

Description:
Causes the Macro Mania window to minimize itself (become an icon). Use **MinimizeAll** to conveniently minimize all Windows programs that are currently running.

Syntax1:
MINIMIZE

Syntax2:
MINIMIZEALL

Notes:
This command is more helpful if used at the beginning of a macro, before a Run or Activate command has been issued. Use the Restore command to return the Macro Mania window to its *Normal* size (use the Run or Activate to restore other Windows).

It is not recommended you use the *MinimizeAll* command with a macro that repeats, as the macro will minimize all the windows every time it repeats, causing the the programs you are using to be minimized and then activated again when you use the Activate or Run commands. In other words, although the macro will run fine, it will cause the macro to appear somewhat turbulent if it minimizes and then activates Windows repeatedly. (You could, however, use the command in a macro that then called a repeating macro with the macro command to prevent that effect.)

It is recommended you use the Minimize command (but not the Restore command) with a macro that repeats because Macro Mania will minimize itself only once and the macro will appear to run smoother.

See Also:
Important Notes for Windows 95 (and above)
Manipulating the State of Windows

Pause

Description:
Use this command to pause a macros execution. Useful if you wish to give yourself time to manually type in some keystrokes or otherwise wish to prevent the macro script from immediately continuing.

Syntax:
PAUSE [n] <show>

n is a number indicating the number of seconds (or fraction of a second) you wish the macro to pause for. The optional word *SHOW* may follow the command to indicate that a small box counting down the number of seconds remaining should be displayed. The display box may be moved, and an option to immediately resume or cancel the macro is also available when the SHOW parameter is used.

Example:
PAUSE 3 show

Notes:
You can also pause for a fraction of a second if needed. For example to pause for just a half second, use .5, a quarter of a second, use .25, etc.

Related Topics:
Important Notes for Windows 95 (and above)
Waiting for another program to finish

Rem

Description:
Use before a command or text so that line is ignored when the macro is executed.

Syntax:
REM <command> or <text>

command or *text* is either a command you do not want to be used in your macro or some text, such as a comment within the macro to help you better identify what the macro is intended to do.

Example:
REM Go to the bottom of the notepad.

Notes:
You may also use a single quote in place of the Rem statement.

Repeat

Description:
Causes a macro to repeat.

Syntax:
REPEAT [n]

n is a number indicating the number of times the macro should repeat.

Example:
REPEAT 3

Notes:
This command should be put on the first line of your macros to help you remember its value. (You may also wish to include its value as part of your macro description.)

You may also use a variable in place of [n] (e.g. REPEAT {var-a})

The repeat option can also be set dynamically by setting the Temporary Repeat Value or using the Repeat_Prompt command.

It is not recommended you use the *MinimizeAll* command with a macro that repeats, as the macro will minimize all the windows every time it repeats, causing the the programs you are using to be minimized and then activated again when you use the Activate or Run commands. In other words, although the macro will run fine, it will cause the macro to appear somewhat turbulent if it minimizes and then activates Windows repeatedly. (You could, however, use the command in a macro that then called a repeating macro with the macro command to prevent that effect.)

It is recommended you use the Minimize command (but not the Restore command) with a macro that repeats because Macro Mania will minimize itself only once and the macro will appear to run smoother.

See Stopping a Macro for strategies to help you stop macros if needed.

Restore

Description:
Use after issuing the Minimize command to make the Macro Mania window normal again.

Syntax:
RESTORE

Notes:
Using the restore command also puts the focus back to Macro Mania. Any keystrokes that are sent with the Send command will be sent to **Macro Mania** unless a Run or Activate command is used to put focus back to another program.

See **Important Notes for Windows 95 (and above)**

Run

Description:
Starts a program or brings an existing program to the front of the Windows environment. (See the notes section below for more details.)

Syntax:
RUN [program], <parameters> <{wait}>

program is the path and name of the executable part of the program. If the path is not included, the current directory, Windows subdirectory, and environment path are searched for a copy of the program.

parameters are any valid parameters the program can accept when started (*be sure to include the comma between the program and its parameters*).

{wait} is an optional parameter that, when included, will notify Macro Mania to halt execution of the rest of the macro until the program that is run is terminated. This option is *very* useful when shelling to programs that run and terminate automatically, such as DOS batch files, etc. (See the first paragraph of the *Notes* below for more details.)

Examples:
This example runs Windows Notepad, automatically opening the file *MYNOTES.TXT*.

```
RUN c:\windows\notepad.exe, mynotes.txt
```

This example runs a batch file and then halts the rest of the macro until the batch file has finished (and terminated itself).

```
RUN c:\sample.bat {wait}
```

Notes:
Be sure to include the special braces ({}) around the wait command when used. (See also Waiting for Another Program to Finish).

Except when the {wait} parameter is used, when the Run command is issued, it first searches your Windows environment and detects if a program is already running. If the program is found, it will pull the current instance of that program to the front, if it is not found it will automatically start a new copy of it for you. (With the {wait} parameter it will always try to start a fresh instance of the program.)

Usually this is exactly what you want; however, there may be times when you do not want to use the {wait} parameter and you would like to start a new copy of a program regardless of whether a copy is already running (this only works with smaller programs that allow themselves to be run more than once). For example, you may want a fresh instance of notepad, regardless of whether you are running another copy of notepad.

If this is the case, by leaving off the path of the program, Macro Mania will not find the previous instance of the program and will run a new copy of it. Note that if the program is not in the current directory, the Windows directory, or in the environment path, you may need to use the CHDIR immediately before the Run command so that the executable or its support files will be found.

If *program* is not found, an error occurs.

Send

Description:
Sends keystrokes to a program as if you had typed them manually. (Issue this command only after you have issued either the activate command or run command.)

Syntax:
SEND [keystrokes]

keystrokes are the keys you would press if you were sitting in a program and typing them manually.

If you want to represent more than one character, append each additional character to the one preceding it. To represent the letters x, y, and z, use *xyz*.

You will be using the Send command a lot. Once you master how to use the Send command to its full potential, you can do virtually anything with your macros! You may wish to print this section for easy reference.

Syntax Details:
The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to the Send command (as noted further below). To specify one of these characters, enclose it inside braces. For example, to specify the plus sign, use {+}. Brackets ([ ]) have no special meaning to the Send command, but you must enclose them in braces as well, because in other programs brackets do have special meaning. To send brace characters, use {{} and {}}.

To specify characters that are not displayed when you press a key (such as Enter or Tab) and keys that represent actions rather than characters, use the codes shown below:

Key	Code
Backspace	{BACKSPACE} or {BS} or {BKSP}
Break	{BREAK}
Caps Lock	{CAPSLOCK}
Clear	{CLEAR}
Del	{DELETE} or {DEL}
Down Arrow	{DOWN}
End	{END}
Enter	{ENTER} or ~
Esc	{ESCAPE} or {ESC}
Help	{HELP}
Home	{HOME}
Ins	{INSERT}
Left Arrow	{LEFT}
Num Lock	{NUMLOCK}
Page Down	{PGDN}
Page Up	{PGUP}
Print Screen	{PRTSC}
Right Arrow	{RIGHT}
Scroll Lock	{SCROLLLOCK}
Tab	{TAB}
Up Arrow	{UP}
F1 - F16	{F1} - {F16}

SpaceBar **{ }** << 1 space between the brackets
Variable **{var-?},** where ? is a letter between a-z or number from 0-99. When using the Send command with variables, unlike with other text and the special commands listed above, you can only issue one variable at a time. However, by either using multiple Send commands or merging the variables, this should impose no problem at all. (See the Let command for more information about using and manipulating variables.)

To specify keys combined with any combination of Shift, Ctrl, and Alt keys, precede the regular key code with one or more of the following codes:

Key	**Code**
Shift	**+**
Ctrl	**^**
Alt	**%**

To specify that Shift, Ctrl, and/or Alt should be held down while several other keys are pressed, enclose the keys code in parentheses. For example, to have the Shift key held down while e and c are pressed, use *+(ec)*. To have Shift held down while an e is pressed, followed by a c being pressed without Shift, use *+ec*. (Note that some applications interpret capital letters as Shift-Letter, which may mean something different to that application. Although this is rare, you may wish to get into the habit of using lower case letters that invoke commands within an application unless you are sure it does not matter to the application whether it is lower case or not.)

To specify repeating keys, use the form {key number} (putting a space between key and number). For example, {LEFT 42} means press the Left Arrow key 42 times; {h 10} means press h 10 times.

The macro will not continue until the SEND command is complete. This feature ensures that the macro does not get ahead of itself. If you are sending a lot of keystrokes, you may wish to break them up into several SEND commands so that the program that is receiving the keystrokes has time to process them before another group of keystrokes are sent.

Limitations: The Send command can not send keystrokes to a program that is not designed to run in Microsoft Windows. Send also can not send the Print Screen (PRTSC) key to any program.

See Also:
Making Sure Keystrokes Get Received.

SendInput

Description:
Sends the most recently entered text retrieved from the GetInput command.

Syntax:
SendInput

Example:
```
GETINPUT Enter the file to open.
RUN notepad.exe
SEND %Fo
SENDINPUT
SEND {enter}
```

Notes:
If no text is in the buffer from the GetInput command, the macro asks if you wish to abort the macro. Answering *Yes* causes the macro to stop, *No* causes the macro to skip the current SendInput command. Like the Send command, an Activate or Run command must be issued before the SendInput command. A GetInput command should also precede the SendInput command.

The SendInput command sends input obtained from the GetInput command if no specific variable was issued to store the input. If a variable was issued with the GetInput command, use the SEND command to send the contents of the specified variable.

SendNow

Description:

Sends the current (system) date and/or time as specified with the *format* parameter. (Like the send command, this command should only be issued after you have issued either the activate command or run command.)

Syntax:

SENDNOW [format]

The following table shows the characters you can use to create user-defined date/time formats (examples are at the bottom):

Character	Meaning
c	Send the date as ddddd and send the time as t t t t t, in that order. Only date information is sent if there is no fractional part to the date serial number; only time information is sent if there is no integer portion.
d	Send the day as a number without a leading zero (1-31).
dd	Send the day as a number with a leading zero (01-31).
ddd	Send the day as an abbreviation (Sun-Sat).
dddd	Send the day as a full name (Sunday-Saturday).
ddddd	Send a date serial number as a complete date (including day, month, and year) formatted according to the Short Date setting in the International section of the Windows Control Panel. The default Short Date format is m/d/yy.
dddddd	Send a date serial number as a complete date (including day, month, and year) formatted according to the Long Date setting in the International section of the Control Panel. The default Long Date format is mmmm dd, yyyy.
w	Send the day of the week as a number (1 for Sunday through 7 for Saturday.)
ww	Send the week of the year as a number (1-53).
m	Send the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is sent.
mm	Send the month as a number with a leading zero (01-12). If m immediately follows h or hh, the minute rather than the month is sent.
mmm	Send the month as an abbreviation (Jan-Dec).
mmmm	Send the month as a full month name (January-December).
q	Send the quarter of the year as a number (1-4).
y	Send the day of the year as a number (1-366).
yy	Send the year as a two-digit number (00-99).
yyyy	Send the year as a four-digit number (100-9999).
h	Send the hour as a number without leading zeros (0-23).
hh	Send the hour as a number with leading zeros (00-23).
n	Send the minute as a number without leading zeros (0-59).
nn	Send the minute as a number with leading zeros (00-59).

s Send the second as a number without leading zeros (0-59).
ss Send the second as a number with leading zeros (00-59).

t t t t t Send a time serial number as a complete time (including hour, minute, and second) formatted using the time separator defined by theTime Format in the International section of the Control Panel. A leading zero is sent if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is h:mm:ss.

AM/PM Use the 12-hour clock and send an uppercase AM with any hour before noon; send an uppercase PM with any hour between noon and 11:59 PM.
am/pm Use the 12-hour clock and send a lowercase AM with any hour before noon; send a lowercase PM with any hour between noon and 11:59 PM.
A/P Use the 12-hour clock and send an uppercase A with any hour before noon; send an uppercase P with any hour between noon and 11:59 PM.
a/p Use the 12-hour clock and send a lowercase A with any hour before noon; send a lowercase P with any hour between noon and 11:59 PM.
AMPM Use the 12-hour clock and send the contents of the 1159 string (s1159) in the WIN.INI file with any hour before noon; send the contents of the 2359 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string sent matches the string as it exists in the WIN.INI file. The default format is AM/PM.

The following are examples of date and time formats:

Format	**What is sent if the current day is December 9, 1995 and the time is 8:50 pm.**
m/d/yy	**12/9/95**
d-mmmm-yy	**9-December-95**
d-mmmm	**9 December**
mmmm-yy	**December 95**
hh:mm AM/PM	**08:50 PM**
h:mm:ss a/p	**8:50:35 p**
h:mm	**20:50**
h:mm:ss	**20:50:35**
m/d/yy h:mm	**12/9/58 20:50**

A Simple Example

Maybe the best way to explain the easy syntax that Macro Mania uses is with an example. Here is an example of a simple macro that will start Windows Notepad and then type the message *Macro Mania is powerful, yet easy-to-use!:*

RUN Notepad.exe
SEND Macro Mania is powerful, yet easy-to-use!

****

For more detailed examples, be sure to review the sample macros that come pre-installed with Macro Mania.

Starting Programs

The first part of a macro needs to start the program that will receive the keystrokes sent from the macro. There are two commands you can use to start a program, either the Run command, or the Activate command.

The method used most often is the Run command to start programs since it can either find a currently running copy and bring it to the front, or run a new instance of the program if it is not found. Another option, though less useful, is to use the Activate command.

Tricks and Techniques

How to Register

This program is distributed as shareware , which means you must pay for it if you continue to use it. Although many people have suggested that I could charge much more, I have set the price relatively low to encourage more payments. Please do not think your payment is not important to me (it is important to me). In fact, there are several benefits I can offer you when you pay for your license(s).

Be sure to see Getting the Latest Version... to make sure you have the latest version of Macro Mania and Upgrade Policy if you have already paid for a previous version of Macro Mania and are upgrading those licenses to a major new release. Also make sure you have reviewed and agree with the License Agreement.

You MUST purchase a license for *each* PC you will use Macro Mania on (multiple licenses may be purchased for a significantly discounted price):

N = Number of licenses being purchased...

1 = $37.50 (no discount)
2-4 licenses (20% discount) = $37.50 x N x .8
5-9 licenses (30% discount) = $37.50 x N x .7
10-19 licenses (40% discount) = $37.50 x N x .6
20+ (50% discount) = $37.50 x N x .5

Examples:
6 licenses = $37.50 x 6 x .7 = $157.50
12 licenses = $37.50 x 12 x .6 = $270.00

For your convenience there are several ways to register:

1) If you have a valid Visa, Mastercard or Discover card, you can contact *NorthStar Solutions* for orders only via any of the following methods (please note that you are registering **Product 1844** and how many copies you need when you contact them so they can immediately give you the appropriate registration code):

INTERNET ORDERS
http://www.nstarsolutions.com/85.htm - fill out the online order form--fast, easy and secure!

PHONED ORDERS
Calls are taken 10 am - 8 pm, EST, Monday thru Saturday.
1-800-699-6395 (From the U.S. only.)
1-803-699-6395
Operators take orders only.

FAXED ORDERS
Available 24 hours.
1-803-699-5465

E-MAILED ORDERS
CompuServe: **starmail**
America Online: **starmail**
Internet: **starmail@nstarsolutions.com**

MAILED ORDERS
You may register with a check or money order.
Make them payable to "**NorthStar Solutions**" and send them to:
PO Box 25262
Columbia, SC 29224
(For fastest delivery of your registration code, provide either an Email address (preferred) or a fax number where

it can be sent. Otherwise one will be promptly sent via snail mail.)

Please provide (or be prepared to provide) the following information:
*** The program you are registering (Product 1844).**
*** Your mailing address.**
*** Your Visa, MasterCard, or Discover # and its expiration date (if using credit card).**
*** Your E-Mail address (so NorthStar Solutions can send you an E-Mail confirming your order and so I can contact you easily with any important follow-up information, upgrade announcements, etc.).**

2) If you have a CompuServe account and would like to register online via CompuServe, you can use their SWREG services to do so.

(For fastest delivery of your registration code, provide either an Email address (preferred) or a fax number where it can be sent. Otherwise one will be promptly sent via snail mail.)

Our Reseller Program

Many consultants/resellers have asked us about a reseller program. These people realize that if people or organizations are introduced to Macro Mania, they will quickly realize the tremendous savings in labor and associated costs that the program can provide, and they will certainly want to purchase it. In other words, Macro Mania is one of those products that appeal to a very wide market and whose features can almost sell itself. With a little effort to introduce Macro Mania to other people and organizations, and the ability to take advantage of our discounts for quantity purchases, a reseller can make a healthy profit with Macro Mania!

Our reseller program permits you to purchase licenses and then resell those licenses in any quantity you wish so long as the total licenses that you resell do not exceed the total licenses that you purchase from NorthStar Solutions. You may use our suggested retail price of $37.50 per license, or you may use whatever pricing schedule you develop to help you generate a profit while selling as many licenses as possible. For example, if you buy 50 licenses and get the 50% discount, then resell those licenses for the suggested retail price of $37.50 per license, you profit $937.50. All licenses purchased by a reseller must be purchased in advance and the following discounts apply to the $37.50 we charge you per license:

Reseller Prices:

N = Number of licenses being purchased...

1 = $37.50 (no discount)
2-4 licenses (20% discount) = $37.50 x N x .8
5-9 licenses (30% discount) = $37.50 x N x .7
10-19 licenses (40% discount) = $37.50 x N x .6
20+ (50% discount) = $37.50 x N x .5

Examples:
6 licenses = $37.50 x 6 x .70 = $157.50
12 licenses = $37.50 x 12 x .6 = $270.00

As a reseller, you will be responsible for a certain level of technical support. This support will be for general installation, any customized macros you want to include in the product you resell, and basic use of the program. NorthStar Solutions will provide lower-level support for the operation of the program, bug fixes, etc. as needed.

All prices and terms subject to change without notice. Contact NorthStar Solutions for the latest information about our Reseller Program using any the following methods (E-mail contact is preferred because it is fast, inexpensive, and accurate. We check Email *several* times a day and can often respond very quickly (and in more detail, if needed).

E-MAIL:
CompuServe - **starmail**
America Online - **starmail**
Internet - **starmail@nstarsolutions.com**

FAX:
(803) 699-5465.

MAIL:
NorthStar Solutions
P.O. Box 25262

Columbia, SC 29224

Upgrade Policy

If you have already bought 1 or more Macro Mania license, all *minor* updates to your license (e.g. v4.0 to v4.1) are free and a 50% discount will be applied to any *major* updates--up to the same number of licenses you have already paid for.

So, if you have bought x licenses for a previous version of Macro Mania, and want to upgrade x licenses to a major new version, you just figure up the current cost for x licenses, then divide that in half and that is the price you pay.

N = Number of licenses being purchased...

1 = $37.50 x .5 = $18.75
2-4 licenses (20% discount) = $37.50 x N x .8 x .5
5-9 licenses (30% discount) = $37.50 x N x .7 x .5
10-19 licenses (40% discount) = $37.50 x N x .6 x .5
20+ (50% discount) = $37.50 x N x .5 x .5

Example 1): You already bought 10 licenses and want to upgrade those 10 licenses:

$37.50	Base Price
x 10	# of licenses to *upgrade*
$375.00	
x .60	40% discount for 10
$225.00	
x .50	50% *upgrade* discount
$112.50	

Example 2): You already bought 10 licenses, want to upgrade those 10 licenses, and then purchase 10 more licenses:

You pay $112.50 as figured above for the 10 *upgrade* licenses, and then just pay for 10 more licenses as follows:

$37.50	Base Price
x 10	# of licenses to buy (not upgrade)
$375.00	
x .60	40% discount for 10
$225.00	

Thus, you buy the first 10 for only $112.50 and the second 10 for $225.00 for a total of $337.50 for all 20 licenses.

(Be sure to see the section called Revision History for an overview of the new features and fixes that have accompanied each new version--you will see that Macro Mania has consistently had features added to both minor and major updates--making sure Macro Mania gets better and better and giving this upgrade benefit significance.)

Need Help? Send e-mail to **starmail@nstarsolutions.com** if you are uncertain the price and would like help figuring it up.

"Shareware"

The essence of shareware is to provide you with reasonably priced software that you get to "try before you buy"--the ultimate guarantee of its quality and usefulness to you. Also, while shareware can be just as "professionally" developed as software that comes in a fancy box, the price of shareware can be set lower because it does not have to cover expensive marketing and advertising costs. Still, some payment is necessary to give shareware developers incentive to continue to develop new products (or improve existing ones), which is why you must pay for this program if you find it useful and continue to use it after a 30-day trial period. Thank you for understanding and honoring the shareware concept.

Technical Support

Contact us using any of the methods listed below, and we will try to help. (Please make sure you have read the Help provided with this program first. The **Important Tricks and Techniques** section has answers to many common questions.)

E-mail contact is preferred because it is fast, inexpensive, and accurate. I check Email *several* times a day and often respond very quickly (and in more detail, if needed).

E-MAIL:
CompuServe - **starmail**
America Online - **starmail**
Internet - **starmail@nstarsolutions.com**

FAX:
The 2nd best way to get help is to fax a scenario of the problem you are having (the more details you can provide the better). We can be reached via **FAX** 24 hours at **(803) 699-5465**.

MAIL:
NorthStar Solutions
P.O. Box 25262
Columbia, SC 29224

Run Command

Starts a program or brings an existing program to the front of the Windows environment.

Activate Command

Brings an already running copy of a program to the front of your Windows environment.

Restore Command

Use after issuing the MINIMIZE command to make the Macro Mania window a "normal" size.

Temporary Repeat Value

Set via the "Settings" pull-down menu in the main form, the Temporary Repeat Value will be set for ALL macros and overrides any internal REPEAT value you may have set using the REPEAT command. You should always make sure a macro is functioning the way you intended before setting any repeat value, temporary or otherwise.

Minimize Command

Causes the Macro Mania command center to minimize itself (become an icon). This is useful if you wish to see what is happening as the macro executes, or plan to immediately start doing something manually in a program after a macro is finished.

Activate Command

Brings an already running copy of a program to the front of your Windows environment. (The RUN command can also do this and is the preferred method because it does this and more.)

Run Command

Starts a program or brings an existing program to the front of the Windows environment.

Quick Start

The easy-to-use commands and syntax for *Macro Mania* will have you creating useful macros in minimal time. The nice thing is that once you learn the 2 important commands of *Macro Mania*, you will have the ability to create macros for *all* your Windows programs!

If you are like most people, you will find using the interface for *Macro Mania* is very intuitive. However, to get a quick start you will need to refer to the few commands that it uses and their syntax in this help section. For now, assume you can refer to them easy enough (because you can). To create a new macro follow these steps:

1. From the main *Macro Mania* form that is presented when you start up, select ***Add/Edit Macros...*** from the *File* pull-down menu, or press *Ctrl-Ins*.
2. From the new form that pops up, select the ***Add*** button.
3. Select **Select Icon** to associate an icon for your macro.
4. **Click once on the icon you want**, then select the ***OK*** button.
5. Move ***(Tab)*** to the description box, then **type in a description for your macro**.
6. Move ***(Tab)*** to the area where you type in the script of your macro.
7. Refer to the *Macro Mania Commands* in this help file for a list of commands available to you. In general, you will probably want to start with the *Run* command to start a program, then use the *Send* command to send keystrokes to your program(s). See A Simple Example for an easy example that you can copy to get acquainted with *Macro Mania*.
8. Once you have typed in your macro, select the ***Save*** button, which will prompt you to Test your macro.

Special Rules for the SEND Command

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to the Send command. To specify one of these characters, enclose it inside braces. For example, to specify the plus sign, use {+}. Brackets ([ ]) have no special meaning to the Send command, but you must enclose them in braces as well, because in other programs brackets do have special meaning. To send brace characters, use {{} and {}}.

SWREG

Please note that this option of registering will cost you $5 more, or $42.50 for one copy, which helps me absorb the 15% of the registration price that CompuServe charges me when someone uses their SWREG service. Quite frankly, with a price as low as it is, I can not afford to take much more out of my small profit per registration.

You can also purchase multiple licenses via SWREG and still get a discount for doing so. However, SWREG does not facilitate a way to apply percentage discounts, so you should contact me first and tell me how many licenses you need and I will send you instructions for using SWREG to pay for multiple licenses.

So why do I offer SWREG as a means of registering? I realize that some of you may find it more convenient to use your CompuServe account to register programs online. I know some of you want to purchase a shareware program with your companys CompuServe account because there is a lot less paperwork involved and approving signatures to obtain, etc.

To register online via CompuServe, first **GO SWREG**, then select **Register Shareware** and move thru the menu-driven system. The registration ID for this program is: **15197**

Once you register online, an Email message will automatically be sent to me notifying me of your registration. I will then promptly (usually within 8 hours and often within only a couple hours if during the day or evening) send you an Email message with the registration code.

Msg_Ok Command

Description:

Stops the macro from executing and displays a message. Execution is immediately resumed when the Ok button is selected.

Syntax:

MSG_OK [message]
or
MSG_OK {var-?}

[message] is any text you wish to have displayed in the message box. You may also display the contents of a variable (see the Let command for more details about setting and manipulating variables).

Notes:

This command is useful for giving the user instructions and/or stopping a macro while another program is executing. If you need the message to be more interactive and allow the option of continuing with the macro or stopping the macro entirely, see the Msg_OkCancel command.

See **Important Notes for Windows 95**

Msg_OkCancel Command

Description:

Stops the macro from executing and displays a message with an OK and CANCEL button. If OK is selected, the macro continues, if CANCEL is selected the macro stops and exits.

Syntax:

MSG_OKCANCEL [message]
or
MSG_OKCANCEL {var-?}

[message] is any text you wish to have displayed in the message box. You may also display the contents of a variable (see the Let command for more details about setting and manipulating variables).

Notes:

The OK button is usually the default button (the button that is highlighted and therefore executed if the enter button or space bar is pressed). To make the CANCEL button the default, extend the command to **MSG_OKCANCEL_2**, making the 2nd button (CANCEL) the default button.

Use the PAUSE command to have a macro pause, but then continue automatically after a default time period has elapsed. The pause command can also display a *RESUME* and *CANCEL* button by using the SHOW parameter, but it can not display a custom message as with the MSG_OKCANCEL command.

See **Important Notes for Windows 95**

Scheduling Macros

The interface for scheduling a macro can be found on the Macro Editor Window and should be easy enough to figure out. Just a couple notes to help you along:

Macro Mania must be running in order for this function to work. It also must be sitting at the regular, main icon window (the scheduler is not activated while you are in the Macro Editor).

You may wish to use the PAUSE command with the show parameter as the first command in any macro that is scheduled so that you can be warned that it is about to begin and you will be able to let it resume or cancel it. At any rate, you will want to stop what you are doing while the macro executes so that you do not interfere with it or otherwise accidently put the focus onto the wrong program and cause very bad things to occur.

Scheduling Times:

Macros scheduled to execute for a specific time interval (hourly, every half-hour, or every 15 minutes) will execute according to your system clock as outlined below:

Hourly: when the minute and second are at 00:00
Half Hourly: when the minute and second are either at 00:00 or 30:00
Every 15 Minutes: when the minute and second are either at 00:00 or 15:00 or 30:00 or 45:00.

Any schedule you set for a macro is saved so that the next time you start Macro Mania that schedule is followed. Use the optional date setting if you only want the macro to run once (rather than every day at the scheduled time).

You can also define a specific time for a macro to run. You are encouraged to use the spin buttons and click the AM/PM buttons accordingly. If you wish to type in the time yourself, do **not** use military time. That is, use the regular time and define the AM or PM setting with the appropriate option button. For example, to have a macro run at 3:10 PM, key in **3** in the hour box, **10** in the minute box, and then select the **PM** option button.

SCHEDULING DAYS:

The interface to schedule days is very easy-to-use (just point-and-click) and can be found by selecting the *day* button from the scheduling interface on the Macro Editor Window. By default, if you only schedule the *time* portion of a macro, the macro will run every day for that time (or time interval) you set. However, you can have macros run on specific days of the week or specific dates by scheduling them to do so. If you select one or more days of the week (e.g. Monday, Tuesday, etc), the specific date (e.g. June 14, 1996) will be ignored. If you wish to have the macro run on a specific date, simply make sure there are no days of the week checked (quickly toggled by unselecting the ALL check box).

Benefits of Registering this Program

1) You will be given a registration code to make your current copy (or copies) fully-registered and legal.

2) No more reminder screens (hurray!).

3) Your name (or your company name) is displayed on the opening screen and in the About window. No more shamefully displaying that you are merely evaluating this program--you will be a full-fledged Macro Mania power user <grin>.

4) You will know in your heart that you have done the right thing (both legally and morally).

5) You will be notified via Email of new releases along with a brief description of new features and any bug fixes.

6) Upgrades: If you have already bought 1 or more Macro Mania license, all *minor* updates to your license (e.g. v4.0 to v4.2) are free and a 50% discount will be applied to any *major* updates (e.g. v4.x to v5.x)--up to the same number of licenses you have already paid for. See, Upgrade Policy for more details.

7) Although technical support has not been needed much, registered users are naturally given more priority and attention regarding technical support.

Revision History

Be sure to see **Getting the Latest Version...** to make sure you have the latest version of Macro Mania.

>> VERSION 4.0 <<

Introduced several Let Commands that enable you to put text, the contents of files, and the contents of the clipboard into variables. You can then parse, evaluate, and change the contents of those variables with variations of the "Let" command.

Other commands such as SEND, MSG_OK, etc will work with variables too!

Added some If-Then commands which enable you to compare variables, check for the existance of a file, and/or check for the existance of a Window and then execute a macro accordingly.

A new GoTo command has also been added so that you can jump to a particular part of a macro (assumedly to use with the If-Then command).

Added several File Commands to enable you to copy, delete, move, rename, and even create/write to files as well as make, remove, and rename subdirectories.

Added the optional {wait} parameter to the RUN command. If you include this parameter, Macro Mania will run the specified program, then halt execution of the rest of the macro until the program has terminated (either automatically or manually). This option is *especially* useful when shelling to programs that run and terminate automatically, such as DOS batch files, etc.

Added the ability for you to Order (resequence) macros. This is useful if, for example, you want to group certain macros together, move hidden macros to the end, etc. This feature will *automatically* detect and adjust any macros that have a macro command so that they point to the correct macro(s) after their sequence has changed.

The macro number is now displayed in the left corner of the status bar--useful to get a visual overview of the macros and their positions when you wish to edit or reposition macros.

Fixed a minor bug with drive box not appearing on high resolution monitors when selecting icons.

Fixed a minor bug where the edit box did not detect a change if you pasted text using Shift-Insert.

Removed the "feature" of the Macro Editor Screen resizing itself according to the screen resolution--this permits the same amount of information to be displayed in the editor, but it does not take up as much screen space.

If you pass in a parameter to tell Macro Mania what macro file to use, you do not have to include the full path. If a path is missing it looks in its current subdirectory, if a path is present it will use the environment path, and if the file is not found at all, then it just uses the DEFAULT.MM file. (See Creating and Opening New Macro Files.)

Added a *Copy Macro* feature to the Macro Editor so you can easily copy an existing macro to a new macro (assumedly to allow you to copy then edit a macro that you may want to be similiar, but not exactly the same as the one you are copying from).

Added a *MinimizeAll* sub-command to the Minimize command, which enables you to quickly and easily minimize *all* Windows currently open.

Added a *Save As* option in the Macro Editor so you can quickly save a macro file to a new file.

Fixed a bug with the passwords for opening or editing macros getting truncated to 5 characters.

Fixed a bug that happened randomly with the ACTIVATE and RUN commands not working when odd situations existed in the Windows environment. Macro Mania has always been a very solid program and is now more solid than ever!

>> VERSION 3.53 <<

Added the OnTop command which enables you to toggle the *Always On Top* property (if the main window of Macro Mania floats on top of the other windows) in a macro.

>> VERSION 3.52 <<

Enhanced the way the windows are displayed on high-resolution monitors.

Added better compatibility with International Date Settings thar are not like *(U.S.)*.

>> VERSION 3.51 <<

Fixed a bug with scheduling a macro between midnight and 1:00 am--it used to not save a macro scheduled in that time range correctly.

Made Macro Mania smarter about saving the size and position of a branch Window when it exits. If it is minimized during the exit it will not save the minimized size and therefore cause the Windows to be virtually unusable. This only happened under unusual exit circumstance with branch Windows, but that has been fixed.

Added some discussion about DOS batch files to the help system.

Added the feature/option to have the first button beep when it gets the focus. This is especially useful if you like to tab through your macros from the keyboard and would like an audio reference of when you have just put focus on (or passed) the first button.

>> VERSION 3.5 <<

Added several commands that enable you to manipulate the contents of the Windows clipboard. Namely the Clipboard_Restore, Clipboard_Save, Clipboard_Set, and Clipboard_SetFile commands.

Added the ability to create different macro files and then open them as needed. Useful if you want to categorize your macros (e.g. by user, functionality, what application they will be used with, etc.).

Added the Open command to open a different macro file from within a macro itself. Also added the ability to pass a macro file at startup and the ability to manually open another

macro file via a selection on the pull-down menu.

Added the Branch command. This command allows you to specify another macro file which will pop up and allow you to interactively select an option. Once an option is selected, a macro will run and return to the original macro to execute further macro commands if needed.

Added the Repeat_Prompt command to give more control and easier access regarding setting repeat values in a macro.

Added the ability to make a macro button invisible from the Main Screen in Macro Mania. This is useful when you do not normally select a macro directly (e.g. it is used via a Macro command in another macro, it is only executed at startup, etc.)

Enhanced Macro Mania so that the macro command no longer has to be the last command of a macro. Execution will now return to a macro that launches another macro. You can now easily create subroutines and loops within macros.

To quickly modify a macro, you can now click once on the *right* mouse button to automatically go to the edit screen to edit the macro your mouse pointer is on. Especially helpful if you have several macros and forget their sequence number.

Added a feature that *automatically* searches and adjusts any macros that have a macro command which points to macros that get resequenced when you delete macros.

Added a few more bells and whistles on the Macro Editor. For example, F3 will take you to the box so you can jump to a macro quickly.

Added a password feature you can set at two levels. You can now password protect your macro file entirely, password protect the macros from being view or edited via Macro Mania, or password protect both.

Added to the discussion of the Run command, which mentions some not so obvious features that will be sure to interest most of you. (See the NOTES section of the Run command for details.)

>> VERSION 3.1 <<

Removed the opening screen if Macro Mania is a registered copy and it is has been run minimized or with a parameter to have it run a specific macro at startup (an unregistered copy requires you to select OK at the opening screen). If Macro Mania is just being run normally (registered or not registered) a Splash Screen is displayed for just a couple seconds when Macro Mania first initializes.

Fixed a minor bug with the scheduler: if a date had been scheduled, one had to select *Not Scheduled*, save it, and then move to another macro or exit the macro setup for it to clear. (Now you only have to select *Not Scheduled*--the need to move to another macro or exit the macro setup is no longer necessary to unschedule a macro.)

Added the ability to schedule macros to run only on certain days of the week (e.g. Monday, Tuesday, etc.).

Added the PlayWav command so that you can play a wav file from a macro for added pizzaz or to otherwise bring attention to a macro. (Kind of a fun to play with too!)

Added the WinExit command which can either exit Windows, restart Windows, or restart the computer--fast and easy.

>> VERSION 3.0 <<

Extended the ability of the Activate command to enable it to activate a program with only a partial match of its Title Bar. This took a lot of extra programming effort (and testing), but I could not ignore the cries of the masses <smile>.

Added the GetInput and SendInput commands. The GetInput command can be used to request small amounts of dynamic information (file names, search strings, etc.). The SendInput command will then send the information obtained from the GetInput command.

Reworked the examples that are installed. Based on the operating system being installed to (Windows 3.x or Windows 95), appropriate examples are installed to best demonstrate the usefullness of Macro Mania.

Enhanced the discussion of Windows 95 considerations in the Help System. (See the Pause or Minimize command for more details.)

>> VERSION 2.2 <<

Added an optional date setting that can be used with the scheduler.

Added 2 more options to select from when toggling button sizes: extra small with icons and extra small without icons. (Now the footprint can be set to an even smaller size if needed.)

Added a new Installation routine (undetectable change for most users).

>> VERSION 2.1 <<

Due to popular demand, added the option to set/toggle the size of the buttons. The buttons can now be smaller (making the program's "footprint" much smaller).

Added the Exit command, which allows you to exit the macro anywhere within the macro.

Added the End command, which allows you to have "Macro Mania" easily end itself entirely. (For those of you who like to just run one macro, perhaps from another program, and then want "Macro Mania" to end.)

>> VERSION 2.0 <<

Added a schedule feature enabling you to schedule macros for specific times or during any 15-minute, half-hourly, or hourly interval.

Added the Msg_Ok and Msg_OkCancel commands that you can use to stop the macro and display a message. You can use either a regular message with an "Ok" button or use an interactive message that displays an "Ok" and "Cancel" button which will resume or cancel the Macro, respectively.

Note: Combine the above two features for a handy alarm/tickler!

Removed the prompt "Exit Macro Mania - Yes/No" when the program is exited.

Added an option to expand the area where you edit the actual macro.

Bug fix: when the window was resized sometimes a few icons would get totally "lost" and the window had to be resized again to "find" them.

Bug fix: when Macro Mania was started minimized the initial screen minimized, but then the main icon window screen would return to its normal state.

Bug fix: when the program was closed from an icon the Window's environment still thought it was running if the task manager or Alt-Tab sequence was used.

Made a few underlying enhancements to fine tune performance.

>> VERSION 1.3 <<

Added the ability to launch a particular macro when "Macro Mania" is first loaded.

Optimized the speed at which macro buttons are adjusted when you resize the "Macro Mania" screen.

>> VERSION 1.2 <<

Fixed a problem that happens on some PC's when the PAUSE command is used.

Removed some duplicate or infrequently used icons that are installed. This keeps the distribution file small. (Search for "Icons" in the Help system to find out how you can easily add your own icons.) Still, there are 200 icons that come with Macro Mania!

Made sure the installation of new versions does not overwrite any macros from older versions unless the users "Okay's it". Too many happy users for me to risk messing that up. <smile>

>> VERSION 1.1 <<

Fixed a small bug that caused the "Always On Top" characteristic to turn off when the program was minimized.

Added the SendNow Command, which allows you to send just about any conceivable combination/format of the current date and/or time to a program.

Added the Beep Command, which allows you to cause the PC to beep via its speaker. Useful when you need to bring attention to the PC, particularly after a long macro and/or when its time for you to type.

(Registration Code)

Given to you after you register your copy (or copies) of Macro Mania, this code will make your copy a fully-registered version.

This code is entered in the "Enter Registration Code" window which can be found under the Help pull-down menu on the main Macro Mania window (the window with all your macros and their corresponding buttons).

It is given immediately after you contact NorthStar Solutions with a valid Visa/MasterCard/Discover, or promptly Emailed to you after you use the SWREG services or send payment by mail.

Macro Mania 95

As tested by literally thousands of Macro Mania users (as well as many of us here at NorthStar Solutions that use Macro Mania daily), Macro Mania works very smoothly with Windows 95. I recently spent quite a bit of time porting Macro Mania code to a 32-bit version and found out some interesting issues: the distribution archive of the program nearly quadrupled in size, the performance was only equal (and in some cases even slower) than the 16-bit version, and the low level API calls were much more cumbersome (opening up the possibilities for bugs--which, knock on wood, has never been a real problem with Macro Mania). I have concluded that, at least at this point, dealing with a 32-bit version of the product just for the sake of saying it is 32-bit is not a very smart choice for any of us. If something is working good, why try to fix it--especially if fiixing it means a bloated distribution archive, no gain in performance (and sometimes an actual *decrease* in performance), and the possibility of introducing bugs that are not there now! For some applications that use a lot of disk I/O, etc., a 32-bit product makes sense because it will help their performance, but for a product like Macro Mania, the bottleneck is not in disk I/O and it makes little sense to port it to a 32-bit product just for the sake of doing so. The only advantage I see at this point would be from a marketing standpoint; but I am a practical person more interested in technical performance, distribution size, etc. than in a lot of marketing hype. I would rather build a better software product than get wrapped up into a lot of marketing hype.

PlayWav

Description:
Plays a *.wav file on any PC set up and equipped with a sound card.

Syntax:
PlayWav [path][wavfile]

where [path][wavfile] is the full path and *.wav file to be played.

Example:
PLAYWAV c:\windows\tada.wav

Notes:
If the wav file and/or path is not found, the macro will simply continue (without playing the wav file).

See Also:
Beep

WinExit

Description:
Allows you to reboot the computer, restart Windows, or exit Windows.

Syntax:
WinExit [n]

where [n] is either 1, 2, or 3 as follows:
1 - reboots the computer
2 - restarts Windows
3 - exits Windows

Example:
WINEXIT 3

Notes:
Macro Mania gracefully shuts down all Windows programs as if you manually closed Windows before closing other programs. If a program can not be closed gracefully, such as a running DOS program, then Windows will not exit and you will be prompted to close the active application before exiting.

Macro Mania is a Windows program and, as a result, Macro Mania is unloaded when this command is run. Therefore, there is no point to have any commands following the WinExit command in your macro, as they will never be executed. (You could, however, put Macro Mania in your Windows Startup Group and pass it a macro number as a parameter to have Macro Mania begin executing a macro immediately upon starting Windows. See Launch a Particular Macro at Startup.)

It may be useful to have a MSG_OKCANCEL command right before the WinExit command so you have the opportunity to abort the macro if desired.

Manipulating the State of Windows:

A Windows program needs to be running either *Normal* or *Maximized* (not *Minimized* as an icon) for it to accept keystrokes. (When you think about it, this is true whether the keystrokes are being sent from Macro Mania, or you are working with a program and typing manually.)

With that in mind, the first set of keystrokes you may wish to send are keystrokes to either Normalize or Maximize a window (to ensure it is not Minimized as an icon and it is ready to accept keystrokes.

The state of most program windows (whether they are Maximized, Minimized, or Normal) can be manipulated by sending keystrokes. By sending *Alt-space* (translated as *SEND %{ }*) with the SEND command, you can pull up the control button of a window and then send the letter you wish to use to either Minimize (n), Maximize (x), or Restore (r) the window. Press *Alt-space bar* now to see this in action (then press *Esc* to get rid of the command button menu). In summary, once you use either the *RUN* command or the *ACTIVATE* command to bring a window to the front, you can use the send command to manipulate its state (which you will want to do if you think the window could be minimized and not able to accept keystrokes).

***Normalize* - SEND %{ }r**
***Maximize* - SEND %{ }x**
***Minimize* - SEND %{ }n**

Since brackets are ignored by the SEND command, note that in the above syntax, the brackets are optional and used to help identify there is a space there.

Launch a Particular Macro at Startup:

You can launch a particular macro when Macro Mania is first started by following the command with the number of the macro you want it to run automatically. For example, to run macro 3 you would use: "[path]MACROM.EXE 3" to automatically start your 3rd macro (where [path] is the location of MACROM.EXE).

Note that while adding new macros will not affect the position of your macros since new macros are simply appended to your macro list, deleting macros in front of existing macros causes them to be resequenced by -1. For example, you have 10 macros and delete the 7th, now the 8th becomes the 7th, the 9th becomes the 8th, and so on. Just bear this in mind whenever you delete macros so you can make any needed adjustments.

Adding Icons to the icon list:

Macro Mania installs 200 icons you can choose from; however, if you want to add to the list of your choices, you can! Just copy any valid icon file(s) to the *icons* subdirectory beneath the subdirectory where you installed *Macro Mania*. (This is done automatically for you if you select an icon from a directory other than the default icon directory.)

There are many icon libraries available and you can probably locate one where you found this program. Be sure to find a library that has actual icons files (*.ico) that can be copied to the icon subdirectory, not icon libraries that are contained in DLLs--unless you have a utility that can extract the icon files from the DLL.

The Macro Mania Icon Selector can read up to 900 icons, so please do not get too carried away <smile>.

Activating Buttons

Most buttons in a program can be activated by sending a space or, perhaps more obvious and common, an {enter}; however, be sure that the focus is on the button you wish to activate. A better way to activate a button is to use its hotkey if it has one. Hotkeys are the Alt-letter combination that invokes that button and can be identified because the letter is underlined. (This discussion is true, at least, if the program follows normal Windows conventions.)

Combine the search feature of the program with a macro

For example, suppose you have 50 expressions that say A = B, but they are not all the same, some say C = D, some say R = Y, etc. Using the *Find* command of your program, you can search for the equal sign (=) and then send the keystrokes that you would do in order to move the value on the left of the equal sign to the right of the equal sign and vice versa.

Here is a macro that would do that, assuming you have already done a search for the equal sign and F3 causes a find next to occur.

```
REM Repeat this 49 times
REPEAT 49
REM Run (or activate) the program (in this case Visual Basic)
RUN c:\vb\vb.exe
REM Search for the next equal sign
SEND {F3}
REM Delete the equal sign and space, then cut what is on its right
SEND {del 2}+{end}%Et
REM Add the equal sign and paste back what was on the left
SEND {home}%Ep=
```

Save Space in Your Macros

Do not use your macros to type every character of a particularly big group of text (several paragraphs). By using the Clipboard_Setfile command, you can quickly put the contents of a large group of text into the clipboard and then paste it where you need. This will also improve performance for the macro in general, as pasting is faster then sending one character at a time with the Send command.

Mouse Movement

First, let me note a significant limitation with mouse movement: it relies on the exact position of a window in order for it to be useful. If a window is resized or moved slightly, the macro becomes unusable (possibly even causing bad things to happen), and either the window has to be repositioned to exactly where it was before or the macro has to be reprogrammed. (Macro Mania is intended to make tasks faster and easier, not vice versa!) The strength and intent of Macro Mania, therefore, is sending *keystrokes*. Since Macro Mania can send any keystroke, you should be able to reliably perform most, if not all, tasks by simply using keystrokes.

A workaround, however, for sending mouse movement under Windows 95 (and above) is to use the *Accessibility Features*, which then allows one to use arrow keys (keystrokes) to perform mouse movement. To access this:

1) Select "Settings" from the Win95 Task Bar
2) Select "Control Panel"
3) Select "Accessibility Options"
4) Select the "Mouse" folder
5) Select "Use Mousekeys"

Here's a short little macro that demonstrates that the arrow keys (down, up, right, left) now move the mouse:

```
RUN notepad.exe
SEND %{ }m
SEND {Down 20}{enter}
```

The above macro should move your notepad window down the screen a little.

I have not found a program that can not be completely manipulated with keystrokes (I think it would have been a short-sighted programmer to create a program that requires the mouse. Perhaps drawing programs are the only exception to the rule that everything should be performable from the keyboard.). If you elect to try using the *Accessibility Feature*, I should mention I have not had enough experience to help you--Macro Mania is advertised to send keystrokes, not mouse movement; but I did want to mention this possible option. Some users have Emailed me and said it has worked nicely for them. :) Just one final note, you will want to take full advantage of Macro Mania being able to send repeated keystrokes without having to retype them for each keystroke (e.g. {down 20} will work the same as {down}{down}{down}....{down} 20 times.)

Getting the Latest Version...

As evident by the revisions, Macro Mania gets new features as fast as time and ideas permit. Macro Mania has been named a "Reviewer's Pick" by professional reviewers at Ziffnet, has been featured as "Program of the Week" from the Windows Users Group Network [WUGNET] on Compuserve, and it has received favorable mention in nationally recognized computer magazines such as *Info World* and *PC World*.

**** THE INTERNET ****
Point you Web Browser to: **http://www.nstarsolutions.com/mm.htm**

**** COMPUSERVE ****
Macro Mania can be located in the Windows Utilities Forum (**GO WINUTIL**). It is in the "Batch/Launch Tools (3)" library as MACROM.ZIP.

**** AMERICA ONLINE ****
The latest version is always made available on America Online, though the way they post shareware to their libraries does not make it easy for me to point you to exactly where you can go (e.g. a keyword) to obtain the latest copy. I suspect by searching for the two keywords *MACRO* and *MANIA* in the software libraries that you should have no problems locating it.

If you are unable to locate it online, send me e-mail and I can let you know what the latest version is and how to purchase an evaluation diskette for a nominal fee to cover Shipping and Handling.

Sending a space (spacebar)

Either of the following two syntax methods will send a space:

Syntax 1: Send << two spaces follow *Send*

Syntax 2: Send { }

Since brackets are ignored in the Send command, the second syntax method is helpful for identifying that a space is there and is a better method.

Stopping a macro

Once a macro is in motion, it is nearly impossible to stop it, which is why I always emphasize saving your work and testing your macros carefully. However, if you do have a macro that you might need to stop (like when you use the repeat command and are not sure of the exact number of times you need it to repeat, etc.) you can explicitly add a command in the macro which would allow you to stop the macro at certain points:

One option would be to place a MSG_OKCancel command in the macro. This would allow you the option to cancel it or press Enter to have it resume.

Another option would be to use the Pause command with the SHOW parameter and a small delay (2-3 seconds), then you would be able to cancel the macro, press resume, or it would count down in a couple of seconds and resume automatically.

For more control over repeating macros, be sure to use the Repeat_Prompt command which allows you to dynamically set how many times a macro repeats.

(Important Commands)

Although Macro Mania sports a variety of commands to add power to your macros, there are really only 2 main commands (RUN and SEND) that you need to be familiar with to create useful macros fast and easy. As you become a more experienced *Macro Maniac*, you can add to your macro vocabulary and create more sophisticated macros. Also note that I have made the syntax as simple and straight-forward as possible, and syntax reference is just a few steps away in this help system.

Clipboard_Set

Description:
Sets the contents of the Windows clipboard with text you specify.

Syntax:
Clipboard_Set [text] or {var-?}

text is a single string of characters you want to put in the Windows clipboard, or you can use a variable that has been assigned a value with the Let command.

Example 1:
CLIPBOARD_SET Let love and faithfulness never leave you; bind them around your neck, write them on the tablet of your heart. - *Proverbs 3:3*

Example 2:
CLIPBOARD_SET {var-33}

Notes:
This command is useful for putting small amounts of simple text strings into the clipboard for future paste commands to other programs. However, using the *Send* command will accomplish this just as effectively, though the information may be sent slightly slower. If a large amount of text, or text which includes carriage returns, is needed, see the Clipboard_SetFile command for a better way to put such information in the clipboard.

If *text* is omitted, the contents of the clipboard is cleared.

The length of the *text* you wish to put in the clipboard can not exceed any limitations for the Windows clipboard.

Clipboard_SetFile

Description:
Sets the contents of the Windows clipboard with the contents of the file you specifiy.

Syntax:
Clipboard_SetFile [file]

file is the full path and file name for a text file that you want put in the Windows clipboard.

Example:
CLIPBOARD_SETFILE c:\windows\readme.txt

Notes:
The file should be a pure text file. Unlike the Clipboard_Set command, this command enables you to also put text with Carriage Returns in the clipboard, since Carriage Returns are copied to the clipboard as they appear in the file.

For this command to work, the length of the file can not exceed 65,535 bytes. Of course, that is probably more than the limit for how much the clipboard can hold anyway. The length of the file can not exceed the limitations of the Windows clipboard for this command to work.

If the file is not found, an error occurs and the macro will terminate.

Clipboard_Restore

Description:
Restores the contents of the Windows clipboard which was saved with the Clipboard_Save command.

Syntax:
Clipboard_Restore

Notes:
This is useful if you do not want the clipboard commands of Macro Mania to permanently overwrite the contents of the Windows clipboard after a macro has finished executing. The Clipboard_Save and Clipboard_Restore commands must be used in the same macro for Macro Mania to associate them correctly.

The Clipboard_Save command must have been used prior to this command, or the conents of the clipboard will be cleared entirely.

Clipboard_Save

Description:

Saves the current contents of the Windows clipboard so that a ClipBoard_Restore command may be used later in a macro to restore the clipboard contents.

Syntax:

Clipboard_Save

Notes:

This is useful if you do not want the clipboard commands of Macro Mania to permanently overwrite the contents of the Windows clipboard after a macro has finished executing. The Clipboard_Save and Clipboard_Restore commands must be used in the same macro for Macro Mania to associate them correctly.

Repeat_Prompt

Description:
Prompts you to enter the number of times you want a macro to repeat. See the repeat command for more information about repeating macros.

Syntax:
Repeat_Prompt <text>

text is an optional parameter you can add to customize the prompt. If *text* is omitted, a generic prompt with the title of the macro will be used.

Example:
REPEAT_PROMPT Enter the number of times we should copy information from Program A to Program B.

Notes:
The prompt will only appear once (the *first* time it is encountered) during the execution of a macro. Like the repeat command, it is generally better to put this command near the top of your macros for easy reference. Unlike the repeat command, however, this prompt allows you to more dynamically set the number of times a macro executes if needed.

The repeat option can also be set dynamically by setting the Temporary Repeat Value, though the Repeat_Prompt command is probably more useful and less dangerous. With the Repeat_Prompt command, the repeat value is retained only while the macro is running and you are reminded to set (or reset) the value as needed. Therefore the Repeat_Prompt command generally supercedes the previous way to set a temporary repeat value.

See Also:
Stopping a Macro
Important Notes for Windows 95

Branch

Description:
Allows you to specify another macro file which will pop up and allow you to interactively select an option. Once an option is selected, a macro will run and return to the original macro to execute further macro commands if needed.

Syntax:
Branch [file]

file is the name of the macro file (or branch file) that you want displayed.

Example:
BRANCH choices.mm

Notes:
Macro Mania will suspend all macro activity until an option is selected from a branch option box. Once an option is selected, the macro for that option is executed and control returns back to the original macro that called the branch. You can cancel a branch by selecting *Exit* from the *File* pull-down menu and the option to cancel or resume the main macro entirely will be presented.

You may nest branch commands from other branch option boxes. Although most Macro Mania commands may be used from a branch, which certainly includes the 2 important commands, the Open command may not be used from a branch option box.

Use the Open command to open a different macro file entirely without just branching to it.

See Creating New Macro Files, for more information about how to create branch files.

As you are testing your Branches, note that their positions and sizes can be manipulated and will automatically be stored for the next time that branch is used (the same feature that is applied to the main Macro Mania screen).

Creating New Macro Files

Saving New Macro Files:
You may find it useful to group your macros into different categories (e.g. by function, user, application they are used for, etc.). Macro Mania allows you to do this by creating new macro files in addition to the default DEFAULT.MM file. Creating new macro files is also necessary if you wish to use the Branch command to provide a branch option box in your macros.

To create a new file, you can either select *New* from the *File* pull-down menu of the Macro Editor and then enter the name for the new file, or save the current macro file you are working with to a new name with the *Save As* option under the *File* pull-down menu. (Note: I recommend you use the MM extension for macro files to keep their extentions consistent. The *Open* feature when you manually select macro files to open will default to looking for all files that end with .MM.)

Opening Other Macro Files:
Although Macro Mania usually tries to locate and execute its default file, DEFAULT.MM, you can pass it the file as a parameter when first running *macrom.exe* for it to automatically open a different macro file when it first loads. For example, macrom.exe c:\macrom\jeff.mm will automatically open jeff.mm rather than DEFAULT.MM when Macro Mania first loads.

Note that you should pass the full path and macro name to tell macro mania where to locate the macro file; however, if no path is found, it looks in its own subdirectory, then if the file is still not found, it opens DEFAULT.MM file. (You can also use the Open command to have a macro open another macro file, or you can open another file manually by selecting Open from the File pull-down menu.)

Open

Description:
Opens another macro file for Macro Mania to use as its main macro file.

Syntax:
Open [file]

file is the name of the macro file that you want to use.

Example:
OPEN jeff.mm

Notes:
See Creating New Macro Files for more information.

Fast access to edit a macro

To quickly access the Macro Editor, just point to the macro you wish to edit from the main Macro Mania screen and click the *right* mouse button. The Macro Editor will appear and the macro you clicked on will automatically be found and ready to edit.

If you have several macros and do not remember their sequence number, you can easily exit the Macro Editor, find the next macro you wish to edit, and repeat this procedure rather than try and search/scroll for them from within the editor.

General Keyboard Navigation

The following are some navigation tips that apply to most Windows programs. Since many Windows program(mer)s follow a set of rules about how a Windows application is supposed to behave, these rules will likely apply (though I obviously have no control if a program(mer) does not follow these conventions). While this is not a definitive discussion on the topic, I think it covers the most common keyboard conventions/shortcuts available in most Windows applications.

Tab - Moves from object to object in a general pattern (often left to right, top to bottom like one would read a book, or in some other logical fashion). To move in the opposite direction, a Shift-Tab combination can often be used. Naturally an exception to this is with large notepad entry fields where a tab needs to act like a tab (usually to quickly indent a certain number of spaces).

Left, Right, Up, Down Arrow Keys - Self explanatory

Ctrl-Left and Ctrl-Right - Usually allows you to jump a word at a time left or right, respectively.

Home and End - Move the cursor to the far left or far right of a field, respectively within an object that holds text. In a list box, it often moves to the very top (home), or very bottom (end) entry.

Ctrl-Home and Ctrl-End - In large notepad entry fields, move the cursor to the top, left corner or bottom, right corner respectively.

Holding the **Shift** key down while moving the cursor within a field has special meaning: it *highlights* the text it crosses similiar to dragging your mouse pointer across text (allowing for subsequent cut, copy, and paste operations). Using the shift key with some of the conventions mentioned above, you can highlight text within a field just as if you had taken the mouse and dragged across the text. For example, to highlight the entire portion of a large text field, you can first move to the top of the field (Ctrl-Home), then while holding the Shift key, move to the very bottom of the field (Ctrl-End). Translated into a macro where a ^ represents the Ctrl key and a + represents the Shift key, it would look like this: Send ^{Home}+^{End}. To highlight just the current line the cursor is on, you would use: Send {home}+{end}

Often letters that are underlined mean that the command can be quickly executed by holding down the Alt key and pressing that letter at the same time. For example, a button or menu that had the word Help can be invoked by holding down the Alt key and pressing H, translated as Send %H in macro syntax.

Other Conventions/Shortcuts:

F1 - Invoke the Help system
Alt-F4 - Quit a program
Ctrl-X - Cut highlighted text
Ctrl-C - Copy highlighted text
Ctrl-V - Paste text currently in clipboard.
Shift-Insert - Another Paste shortcut
Del - Delete character to the *right* of the cursor (or a currently highlighted section)
BackSpace - Delete character to the *left* of the cursor (or a currently highlighted section)
PgUp - Scroll one screen up
PgDown - Scroll one screen down

Alt-Space - Display the control box of a Window

Just a final note about keyboard shortcuts: programs often have their own shortcuts built into them, and you are encouraged to take advantage of them if you can. For example, in word processors, etc. Ctrl-B often means to toggle the **bold** attribute of the font, Ctrl-U to toggle underline, Ctrl-I for *italic*, etc.

Password Protecting Your Macros

To offer *some* security over your macros, you can define a password that must be used to either open a macro file, view/edit you macros via Macro Mania, or both. This password protection scheme is not hack proof in that someone with reasonable knowledge of computer files can view and/or edit your macros, but it does offer protection from casual nosiness or prying eyes.

Actually, the only reason it was added is because some have mentioned they would like to create macros for others without the others going in and messing them up, in which case you can set the password level to prevent someone from accessing the Macro Editor screen. To prevent anyone but people with the password to be able to open (and consequently run) macros, you can password protect the macro file entirely.

The password and level of use is set up via the *Set Password Options* found under the File pull-down menu on the Macro Editor.

IMPORTANT NOTE: As mentioned above, this password protection scheme is not absolutely hack proof, but should offer some reasonable level of protection for preventing people from easily viewing, editing, and/or using your macros. However, if you use Macro Mania to send passwords, etc. you should note that someone with a little more than casual experience with Windows may be able to view your macros via a way other than using Macro Manias interface. If you think that information in your macros could become compromised, you should not hard code that information into your macros. Consider using the GetInput and SendInput commands in your macros to dynamically send password information, etc. from your macros.

Subroutines within macros (looping, etc.)

There may be times when you want only a certain portion of a macro to loop. That is, you have something that needs to be done repeatedly, but you do not want the whole macro to be done repeatedly. You can do this easily with Macro Mania by combining a couple of its features:

Create a separate macro that does what you need to be done repeatedly. Then by using the Repeat (or Repeat_Prompt) command, you can control how many times that part repeats. You are essentially creating a looping subroutine. Then simply use the Macro command to call that repeating macro, and only that portion will repeat, not the entire macro. If you are concerned about the repeating macros getting in the way, etc. you do not need to be--just make them Invisible by checking that option in the Icon box when you set them up. That way theyll be available to your other macros, but not available (or in the way) as a selection from the Macro Mania Main Screen.

Also, since your macros are *automatically* adjusted when you delete or reposition macros, you will not have to worry about keeping track of their sequence, etc.--it is all done for you. (See Resequencing Macros.)

DOS Batch Files

Sometimes there is nothing as good as a DOS batch file for accomplishing what you need to do. With DOS batch files, you have a whole arsenal of commands that allow you to copy, delete, and move files, make and remove subdirectories, etc. DOS batch files can branch (goto), receive and/or check for conditional parameters, and much more. The commands in DOS are quite powerful and, honestly, much more tried and tested than the file commands I have included with Macro Mania. DOS has been around for a much longer time and has a *lot* of development time already invested in it. Just as an example, go to a DOS prompt and type xcopy /? to get an idea of all the features and parameters for just that one command!

Batch files can check for the existance of files (*if exist* and *if not exist*), display prompts and solicit user input (see the DOS *Choice* command), run programs, etc. This help section is not intended to be a DOS tutorial, but I do want to point out this often overlooked resource--which can easily be used in combination with macros to accomplish those kinds of tasks.

To use DOS batch files with Macro Mania, simply create your batch file and then use the Macro Mania Run command to launch the batch file. Use the optional {wait} parameter with the Run command to halt execution of the macro until your batch file has finished.

So why have I included some DOS-like commands in Macro Mania? One big, ugly limitation of the DOS interface is it is still a DOS interface. That is, you lose the true *Windows look-and-feel* when shelling out to DOS, especially within a fancy Windows macro. Also, some people entered the computer scene after the DOS prompt was something usual to work with and may not be as familiar with DOS as others. However, if you are not afraid of the DOS prompt and need the power, do not look over the tried and true commands that come with DOS. The commands I have included with Macro Mania have been moderately tested and clearly work just fine for modest needs, but they should be tested in any of your macros thoroughly to be sure they meet your needs.

Helpful Hints for Beginners

Practice with a few simple macros:
Macro Mania is very powerful and relatively simple to use. The nice thing is that once you master it, you can use it for *any* windows program without having to relearn new syntax for every program! However, you may wish to start off gradually, creating a few simple macros to get an understanding of the basic concept. Also note that there are only two important commands that are truly essential to creating useful macros with Macro Mania.

Get familiar with the SEND command:
The Send command is the core part of Macro Mania, offering a full host of options to ensure you can do virtually anything, send any keystroke, and be as productive as possible without a lot of repetitive typing. I recommend that if you print anything, you print the syntax of the *Send* command in this help screen so you will have easy access to the syntax while you create your macros. Be sure to pay careful attention to the special rules of the *Send* command.

Save your work:
Before executing a macro, especially a new one, save your work in all your running applications so that if a macro you write does something you did not quite expect, you can return to the original state of the program easily.

Manually perform the steps and write them down:
Before trying to create a macro, perform the task manually and carefully record every key that you press. Another way to write macros is to perform a few keystrokes, switch over to Macro Mania to record them into your macro, then switch back and perform a few more keystrokes, switch back, etc.

Debugging your macros:
Macro Mania sends keystrokes fast--sometimes too fast for you to detect where a macro may be doing something unexpected or missing a step. For debugging: if you are not sure where your macro is not working correctly, place the exit command a few lines above where you think the error is happening. Then gradually move it down one line at a time until the line with the error is identified. (It may also be useful to break your SEND commands into smaller ones for this purpose.)

Make macros that are easy to read and follow:
Macro Mania allows you to put in extra spaces, comments, capitalize its key words, and issue commands repeatedly--so try to make your macros easy to read. For example, while the following two macros do the same thing, one is obviously easier to follow than the other:

EXAMPLE 1:
```
MINIMIZE
RUN WRITE.EXE
send %Pc^B%CeJeff Camino{enter}P.O. Box 25262{enter}Columbia, SC  29224{enter
3}^{home}+{end}^I^{end}{F5}
```

EXAMPLE 2:

```
MINIMIZE
RUN write.exe

REM Center the text, make it bold, then enlarge the font
SEND %Pc^B%Ce

REM Type in the letter head
SEND Jeff Camino{ENTER}P.O. Box 25262{ENTER}Columbia, SC   29224{ENTER 3}

REM Go back and italicize my name
SEND ^{HOME}+{END}^I

REM Go to bottom, make text normal and left justify paragraph
SEND ^{END}{F5}%Pl
```

Always test your macros:

Test your macros, especially before issuing the MACRO command that calls other macros, or the REPEAT command which causes the macro to repeat itself. If a macro is not working properly, you definitely want to find out BEFORE you have issued the macro several times.

Remember, once you master using Macro Mania, it is the only syntax you will need to remember to create macros for many, if not all, of your Windows programs!!

Recording Keystrokes

Macro Mania does not record your keystrokes as you enter them. Still, translating your keystrokes into a macro is *very* easy and this should not present any problem. When I create a macro I either remember the keystrokes needed, or if it is a long sequence, I simply start editing the macro, then flip over to whatever application is getting the keystrokes and perform a few, flip back to the editor to write the macro reflecting what I have typed, etc. The process is really easy and takes very little time. Once you have done this a couple times you will appreciate how easy it is, and that macros can be edited, portions copied and reused for other macros, etc. As you become more experienced, you will be creating and editing macros very quickly and easily.

Launching Macros

To launch a macro, simply click once on the button or use the tab key on your keyboard to tab to the macro you want, then press Enter or Space to launch it. The description of the macro will be shown in the bottom bar of the main Macro Mania Window when that button is highlighted (not necessarily chosen). You should use descriptions to best identify what a macro does and you can further aid yourself by selecting icons to match.

Another option is to assign a *hotkey* to the button which can be done when you select the *Use a Hotkey* button at the bottom of where you select the icon/hotkey for the macro. While not as graphical as icons, hotkeys enable you to quickly access a macro button by enabling you to just press *Alt-character.* For example, if you assign the letter A, you can then just press Alt-A from your keyboard and the macro will be executed without you having to highlight it with the mouse or tab to it via the keyboard.

IMPORTANT NOTE: Macro Mania does not intercept keystrokes being sent to other programs and, therefore, needs to be activated in order to run a macro. I could have added a "keyboard hook" that would trap, interpret, and perform keystrokes being sent to every Windows program, but this is not an easy task and causes a great deal of overhead because, when you think about it, there is a program layered between you, your applications, and *every* keystroke you send (even keystrokes from Macro Mania itself) and with the need for speed, this would be rather cumbersome. However, a quick Alt-Tab over to Macro Mania and then invoking a hot-key you have assigned is still many less keystrokes than usual if you have macros that do anything substantial.

Related Topics:
Launch a Particular Macro at Startup
Adding Icons to the icon list

Resequencing Macros

When you add macros, the new macro is just appended to the end of the macro file/sequence. However, you may find it useful to sequence your macros in a certain order. For example, you may wish to put macros in a logical order, group like macros together, and/or put all hidden macros at the end of the sequence.

To resequence a macro, just determine what new position you want to put it in (use the macro number indicator at the top of the editor screen or on the status bar display of the main window), then go to the macro you wish to resequence, select the *Order* button and enter the new position you want it to be at.

This will not only put the macro in the new position, but it will *automatically* adjust any macros that use the macro command. Otherwise they could end up pointing to the wrong macro if a macro they point to is resequenced to another position (and, thus, assigned a different macro number.

Waiting for another program to finish

If you use the {wait} parameter with the Run command, Macro Mania will halt executing the rest of the macro until the program has terminated (either by itself automatically or manually by a user). This is especially useful with batch files, etc. that you want to run and that terminate themselves automatically.

IMPORTANT NOTE: To accomplish the above feature, Macro Mania first determines the number of applications currently running right before it launches (Runs) the program, then after it RUNs the new program, it sits and waits until that number is the same again before continuing. Thus, if any program is terminated, Macro Mania will begin executing the rest of its macro. Normally this will not be any problem, but you should just be aware of how it works so that you do not inadvertently close any other application while Macro Mania is in its *wait* state. Also note that this feature will not work if the RUN command merely activates a program that has been running already. That is, the RUN command needs to launch the program that the macro is waiting on as a new instance. (If needed, you could ensure the program is closed using the If ExistWindow and issuing the commands to close the window, then RUN it and, thus, this would make sure it was run new, not just activated to the front.)

Activating child Windows

Child Windows are those small Windows that are contained within the larger (parent) window of a program. You can activate these smaller windows by using a Alt-[minus sign]. Translated in Macro Mania syntax as follows:

SEND %-

Note that you may also tab through the child windows by sending F6 (SEND %{F6}) or sometimes by using the arrow keys once the control box menu (the little pull-down menu in the top, left corner) has been activated with either Alt-Alt-[minus sign] or Alt-[spacebar].

Related Topics:
Manipulating the State of Windows

OnTop

Description:
Enables you to toggle the *Always On Top* property (if the main window of Macro Mania floats on top of the other windows) in a macro.

Syntax:
OnTop = [True/False]

Example:
OnTop = True

Notes:
Especially useful if need to watch what is happening in a macro and then need to manually intervene to make the macro continue via a MSG_OK or MSG_OKCANCEL message box. By setting OnTop = True, the message box will float on top so that you can watch what is happening and yet be ready to select the appropriate item in the message box when ready without having to Alt-Tab back to Macro Mania.

Activating the Windows 95 Desktop

In order to activate the desktop, you need to RUN explorer (be sure to include the full path as in the example below) and then you can navigate using arrow movement, home, end, etc to get where you need. This is useful if you need to, for example, get to the Dial-up Network built into Windows 95 because it does not seem to have a standalone EXEcutable file that one can just run using the RUN command. So, heres an example of how I would access the Dial-up Network on my system, which may vary from system to system depending on the location of folders, etc.:

```
RUN c:\windows\explorer.exe
SEND {home}
SEND {enter}
pause 1
SEND {end}
SEND {enter}
pause 1
SEND {home}
SEND {enter}
```

Note that I have added a few strategic pauses to accomplish this, as apparently this procedure (if done via a macro), is too fast for Windows 95 Explorer--the things we go through to get around the barriers constantly thrown at us <sigh>.

Clipboard Commands

The following commands enable you to quickly and easily retrieve and set text into the Windows clipboard.

Clipboard_Set

Clipboard_SetFile

Clipboard_Restore

Clipboard_Save

Related Commands:

Let (ClipBoard)

If-Then

Description:
Allows conditional execution of a macro based on evaluation of an expression.

Syntax:
If *condition* **Then** *ThenCommand* [**Else** *ElseCommand*]

* *Condition* may be the comparison of two variables such as:
IF {var-a} **=** {var-b} THEN...
IF {var-c} **<>** {var-d} THEN...
IF {var-e} **>** {var-f} THEN...
IF {var-g} **<** {var-h} THEN...
IF {var-g} **=>** {var-h} THEN...
IF {var-g} **=<** {var-h} THEN...

* or *condition* can check for the existance of a file as follows:
IF **ExistFile** [FileName] THEN...
where [FileName] is the path and file to check for.

* or *condition* can check for the existance of a Window as follows:
IF **ExistWindow** [Title Bar] THEN...
where [*Title Bar*] is the caption that is found at the top of the programs windows (like the *Macro Mania Help* title bar shown above).
The *Title Bar* must match exactly unless you use **ExistWindow2**, which will find partial matches. Neither command requires the *Title Bar* to match the case (upper or lower case).

* or *condition* can check for the existance of a string (another variable) within a variable as follows:
IF {var-?} **Contains** {var-?} THEN...

* *ThenCommand* and *ElseCommand* are any valid Macro Mania commands (except another IF-THEN command). The Else part is optional. When omitted and the condition is *false*, the macro just continues to the next macro line.

Examples:
```
IF {var-b} = {var-c} THEN GoTo FINISH
IF {var-c} > {var-d} THEN BRANCH c.mm ELSE BRANCH d.mm
IF {var-r} <> {var-s} THEN EXIT
IF {var-1} => {var-2} THEN GoTo CheckVar3 ELSE exit
IF {var-3} =< {var-4} THEN exit
IF ExistFile c:\autoexec.bat THEN BRANCH edit.mm
IF ExistWindow NorthStar - Orders THEN GoTo NS
IF ExistWindow2 NorthStar  THEN GoTo NS
IF {var-s} Contains {var-19} THEN goto Kansas
```

Notes:
The *equal* (=) and *not equal* (<>) conditions compare the variables as *strings*, which means you can compare text or numbers. However, note that, for example, 1,234 <> 1234. Use the Let (Number) command to convert strings to numbers. (Note, however, the *greater than* (>), *less than* (<), *equal or greater than* (=>), and *equal or less than* (=<)

imply you are evaluating numbers and you do not need to convert the strings to numbers when using any of those comparisons. If a string does not have any number in it, the string will equal 0)

When using *equal* (=) and *not equal* (<>) conditions, the comparison is case sensitive; thus, *NorthStar <> Northstar*. If you do not want the comparison to be case sensitive, first set the variables to all one case using either the Let (UCase) or Let(LCase) command, then make the comparison.

The *greater than* (>), *less than* (<), *greater than or equal* (>=), and *less than or equal* (<=) conditions compare numbers only, which means, as expected, 1,234 > 1233, 1234 < 1,235, 1.09 < 1.30, 123 <= 123, 123 >= 122.9, etc.

Be careful when comparing strings. Mainly, be careful that the string you are comparing is what you think it is. For example if you highlight a line of text in a word processor by going to the beginning of the line and then use the down arrow, it is possible that you will also get a Carriage Return, Line Feed (CR, LF) as part of the text. This is different than highlighting a line of text by going to the beginning of the line and then to the end of the line (not just dropping down). You can confirm this by experimenting yourself and seeing the results of when you paste the text somewhere. The first method will paste the text and then drop down to another line while the second method will leave the cursor at the end of the text. So, if you are comparing a line of text and do not want the CR, LF, you should use the second method of highlighting a line of text.

If you are using the *Contains* condition, the search *is case sensitive*, so if the string contains hello and you search for Hello, the condition will not be true. If you wish for the search to not be case sensitive, just use either the LCase or UCase sub-command of the LET command to make both variables in the command all one case before issuing the comparison.

The command must all be on the *same* line (or naturally wrapped around with the word-wrap). In other words, *do not press enter to break up the If-Then command*.

As noted in the syntax, the *ThenCommand* or *ElseCommand* may be any valid Macro Mania command except another If-Then command. If you need to check for a condition and then check for another condition, just use the Goto command to branch to another If-Then command in your macro.

Note that except for the *ExistFile* and *ExistWindow(2)* conditions, the *condition* part of the command uses variables. You need only use the *Let* command previously in your macro to make those variables mean something, such as setting a variable equal to text, the contents of a file or the clipboard, etc.

The *ExistWindow(2)* is useful to first check for a Window that may indicate if you need to use the RUN command and then send keystrokes to close the program. If the program is not running, there is no need to run it just to close it again.

The IF-THEN command can bring a great deal of power to your macros. Programmers will appreciate the fact you can now write flags to files, then read in those flags with a macro and then execute the macro accordingly. In fact, you could use a basic flag of just creating a file, any file, and then using the ExistFile condition noted above. If a file exists, the macro does one thing, if it does not exist, the macro does another thing. To take it one step further, you could put the file into a variable using the Let (File) command. Then you can evaluate or search for an expression in the variable with this If-Then command and have the macro act accordingly.

Related Commands:

While you can use any valid Macro Mania command with the IF-THEN command (except another IF-THEN command), the following commands are likely to be used most often:

Let
Goto
Branch
Exit

GoTo

Description:
Causes execution of the macro to jump to another place (label) within the macro.

Syntax:
GOTO [label]

where [label] is a valid label in the macro as defined below.

Example:

```
LET {var-b} = {1234567}
 LET {var-c} = CLIPBOARD
 IF {var-b} = {var-c} THEN GoTo FINISH
 .
 .
 :FINISH
.
.
Msg_OK Macro is now complete.
```

Notes:
A label can be any alphanumeric string and always begins with a single colon (:). Spaces are also permitted.

Do not include the colon (:) in the GoTo command, only include the colon at the beginning of the label itself.

The [label] is not case sensitive, so :LABEL 32 = :Label 32.

If [label] is not found, an error occurs and the macro stops.

As shown in the example above, the GOTO command is especially useful when combined with the IF-THEN command.

Let

Description:
This command has several sub-commands that enable you to save text to variables and to manipulate the contents of those variables.

Sub-Commands:

Setting a variable equal to something:
- Let (Equals)
- Let (ClipBoard)
- Let (File)

Changing the case:
- Let (LCase)
- Let (UCase)

Trimming off extra leading and trailing spaces:
- Let (LTrim)
- Let (RTrim)
- Let (Trim)

Extracting parts of a string:
- Let (Left)
- Let (Right)
- Let (Mid)
- Let (Remove)

Concatenating/Merging:
- Let (Merge)

Converting:
- Let (Number)

Other Commands that work with Variables:
Send
Clipboard_Set
GetInput
If-Then
Msg_OK
Msg_OKCANCEL
Repeat
WriteFile

Let (Equals)

Description:
Set the contents of a variable equal to the contents of another variable or text.

Syntax:
LET {var-?} = {var-?}
or
LET {var-?} = {text}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-?} = {Macro Mania saves me a lot of time! }

Notes:
The special brackets ({}) must be used when identifying variables.

The special brackets ({}) are optional when using text, but are needed if the text is supposed to have leading or trailing spaces (otherwise the spaces are automatically trimmed off). The special brackets are also necessary if, by some chance, you wanted to use any of the special cases, such as letting a variable equal the word *CLIPBOARD*, not the special meaning it has with the Let (Clipboard) sub-command. Thus, LET {var-?} = CLIPBOARD is not the same as LET {var-?} = {CLIPBOARD}. The first example will make it equal the *word* clipboard, the second example will make it equal the contents of whatever text is in the Windows clipboard.

See Also:
Important General Notes for the Let Command.

Let (ClipBoard)

Description:
Sets the contents of a variable equal to whatever text is in the Windows clipboard

Syntax:
LET {var-?} = CLIPBOARD

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-55} = CLIPBOARD

See Also:
The various clipboard commands for ways to set the contents of the clipboard.

Important General Notes for the Let Command.

Let (File)

Description:
Sets the contents of a variable equal to the contents of a file.

Syntax:
LET {var-?} = FILE [FileName]

where [FileName] is the path and name of the file to read.

Example:
LET {var-f} = FILE c:\windows\readme.txt

Notes:
If the file is not found, the contents is set to null. Use the *ExistFile* condition of the IF-THEN command if you need to check for the existance of the file first.

The length of the file can not exceed 65,535 bytes.

See Also:
Important General Notes for the Let Command.

Let (LCase)

Decription:
Changes the case of all characters in a string to lower case (no capital letters).

Syntax:
LET {var-?} = LCASE {var-?}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-8} = LCASE {var-8}

Notes:
You can let another variable equal the new value, or just manipulate the contents of the same variable as shown in the example above.

See Also:
Important General Notes for the Let Command.

Let (UCase)

Description:
Changes the case of all characters in a string to upper case (all capital letters).

Syntax:
LET {var-?} = UCASE {var-?}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-8} = UCASE {var-8}

Notes:
You can let another variable equal the new value, or just manipulate the contents of the same variable as shown in the example above.

See Also:
Important General Notes for the Let Command.

Let (LTrim)

Description:
Trims off any extra spaces on the left side of a variable.

Syntax:
LET {var-?} = LTRIM {var-?}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-8} = LTRIM {var-8}

Notes:
You can let another variable equal the new value, or just manipulate the contents of the same variable as shown in the example above.

See Also:
Important General Notes for the Let Command.

Let (RTrim)

Description:
Trims off any extra spaces on the right side of a variable.

Syntax:
LET {var-?} = RTRIM {var-?}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-8} = RTRIM {var-8}

Notes:
You can let another variable equal the new value, or just manipulate the contents of the same variable as shown in the example above.

See Also:
Important General Notes for the Let Command.

Let (Left)

Description:
Returns part of a string from the left side of a variable and the number of characters specified.

Syntax:
LET {var-?} = LEFT {var-?} n

where ? is any letter from a to z or any number from 0 to 99, and *n* is the number of characters to read beginning at the far left side of a string.

Example:
LET {var-t} = LEFT {var-c} 145

Notes:
If n exceeds the length of the contents in a variable, all characters are returned.

See Also:
Important General Notes for the Let Command.

Let (Trim)

Description:
Trims off any extra spaces on both sides of a variable.

Syntax:
LET {var-?} = TRIM {var-?}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-32} = TRIM {var-32}

Notes:
You can let another variable equal the new value, or just manipulate the contents of the same variable as shown in the example above.

See Also:
Important General Notes for the Let Command.

Let (Right)

Description:
Returns part of a string from the right side of a variable and the number of characters specified.

Syntax:
LET {var-?} = RIGHT {var-?} n

where ? is any letter from a to z or any number from 0 to 99, and *n* is the number of characters to read beginning at the far right side of a string.

Example:
LET {var-c} = RIGHT {var-c} 75

Notes:
If n exceeds the length of the contents in a variable, all characters are returned.

See Also:
Important General Notes for the Let Command.

Let (Mid)

Description:
Returns part of a string from a variable from any given starting point and optionally reads a certain number of characters.

Syntax:
LET {var-?} = MID {var-?} start [length]

where ? is any letter from a to z or any number from 0 to 99, and *start* is the starting position and [length] is the optional number of characters to read.

Example:
LET {var-g} = MID {var-c} 3 10

Notes:
If [length] is omitted or exceeds the number of possible characters to return, all characters beginning from the *start* position are returned. The start value should be equal or greater than 1, if it exceeds the length of the contents in a variable, no characters are returned.

See Also:
Important General Notes for the Let Command.

Let (Remove)

Description:
Removes a specified number of characters from the *right* side of a string

Syntax:
LET {var-?} = REMOVE {var-?} n

where ? is any letter from a to z or any number from 0 to 99, and *n* is the number of characters to remove.

Example:
LET {var-d} = REMOVE var-c 10

Notes:
If n exceeds the number of possible characters to remove, all characters are removed. You can use the Let (Mid) sub-command to remove characters from the left side of a string.

See Also:
Important General Notes for the Let Command.

Let (Merge)

Description:

Concatenates the values of 2 or more variables into a single variable string.

Syntax:

LET {var-?} = MERGE {var-?}+{var-?}+{var-?}+...

where ? is any letter from a to z or any number from 0 to 99.

Example:

LET {var-g} = MERGE {var-44}+{var-45}+{var-46}

Notes:

This sub-command is especially useful if you want to write several variables to a single line with the WriteFile command, since the WriteFile command will automatically add a Carriage Return, Line Feed when writing a file.

_____

See Also:

Important General Notes for the Let Command.

Let (Number)

Description:
Returns only the numbers in a string.

Syntax:
LET {var-?} = NUMBER {var-?}

where ? is any letter from a to z or any number from 0 to 99.

Example:
LET {var-c} = NUMBER {var-c}

Notes:
This sub-command is useful if you wish to use the If-Then command to compare a numeric string with another numeric string and want, for example, 1,234 = 1234.

See Also:
Important General Notes for the Let Command.

Important General Notes for the Let Command

Be sure to use the special brackets ({}) to identify the variable. This also applies to when variables are used in other commands. For example, SEND {var-32} or Msg_OK {var-h}.

Use a space on each side of the equals (=) sign when assigning values to a variable. This is necessary for the syntax to work correctly and makes the macro easier to read. Thus, you should use LET {var-a} = Hello rather than LET {var-a}=Hello.

Variables must follow the format presented in the syntax ({var-?}), where ? is any letter from a to z or number from 0 to 99. The a-z part is not case sensitive and uses the 26 letters of the U.S. alphabet. Thus, 126 variables are available.

Except for the *Let (Equals)* sub-command, all the other sub-commands use the text within a *variable*, not text itself. If needed, first put text into a variable with the *Let (Equals)* sub-command to later use it with any of the other sub-commands.

The contents of the variables are kept in memory as long as Macro Mania is running and are carried acrossed macros and even across branches and other macro files when opening new macro files. If you need to clear the contents of a variable (or want to be sure it is clear), let the contents equal blank (e.g. LET {var-a} = { }).

File Commands

Copy

Delete

MkDir

Rename

RmDir

WriteFile

Related (file) Commands:

- Let (File)
- Clipboard SetFile
- IF ExistFile... THEN

Copy

Description:
Copies one or more files.

Syntax:
COPY </y> [Source Files] TO [Destination]

where </y> is optional and tells Copy to automatically overwrite any existing file in the *Destination*, *Source Files* is the full path and file specification (similiar to the DOS copy command, (wildcards such as *, ?, and ~? are permitted.), and *Destination Directory* is the full path to the subdirectory where *Source Files* should be copied to.

Example:
COPY c:\dir1*.txt TO c:\dir2

Notes:
See DOS Batch Files for other options.

Always include the file pattern (even if just *.*) for the *Source Files* parameter.

If copying more than one file (e.g. you use any wildcards such as *, ?, or ~?), be sure to include only the directory for the *Destination*.

Remember to include the TO part of the command. (It is a required part of the command and it helps break up the command visually, especially when a file/directory has spaces in its name.)

If the *Destination Directory* is not found, you will given the option to Abort the macro, have Macro Mania try to make the directory and then Retry the copy operation, or Ignore the error and continue with the rest of the macro. If you are not sure whether a directory already exists, use the MkDir command to automatically make the directory and to avoid this error.

If *Source Files* is not found, an error occurs and you are given the option to Abort the macro, Retry the copy operation, or Ignore the error and continue with the rest of the macro. Use the *ExistFile* condition in the If-Then command if you want to avoid this error message entirely.

The Copy command does not currently support long file names (at least entirely). However, you may use a ~? to represent the 7th character and beyond for all files with 9 or more characters in their prefix name (where ? is a number the operating system has assigned to represent the truncated portion of the name). For example to copy a file called *thisismyfile.tmp* from *c:\docs* to *c:\test*, use Copy c:\docs\thisis~1.tmp to c:\test. (Note: If there is more than one with a long name, the 7th position and beyond might be represented with a ~2, ~3, etc.). You may, however, use a wildcard to get all long file names. For example, to ensure all long files beginning with *thisis* are copied, issue the command Copy c:\docs\thisis~?.* to c:\test

Macro Mania does not have a *Move* command. To move a file, simply issue the Copy command and then the Delete command, or use the Rename command which can also move a file on the same drive.

If the number of files to be processed is more than 100, or the sum of the file sizes is greater than 1MB, or either the *Source Files* or the *Destination* is on drive a or b, then a

status window will display during the operation (since the operation is likely to take a couple seconds longer if one of those conditions are met).

WriteFile

Description:
Writes/Appends the contents of text or a variable to a file.

Syntax:
WriteFile [file] {var-?}
or
WriteFile [file] {text}

where *file* is the full path and file name for a file that you want to write to, {var-?} is any valid variable you have set using the Let command, and {text} is any text you wish.

Example:
WriteFile c:\test.txt {Columbia, SC 29224}

Notes:
See DOS Batch Files for other options.

WriteFile will automatically create [file] if it does not already exist, or if it exists it will append directly to it. If you need to make sure the file is new, you may delete it first using the Delete command.

WriteFile automatically adds a Carriage Return, Line Feed (CRLF) to the end of the string it writes to the file. To write one contiguous string on the same line, use the MERGE sub-command of the Let command to first concatenate the string before writing it to the file.

If the *File* to be written to is on drive a or b, then a status window will display during the operation (since the operation is likely to take a couple seconds longer when writing to a diskette).

Delete

Description:
Deletes a file.

Syntax:
Delete [file]

where *file* is the full path and file name for a file that you want to delete.

Example:
Delete c:\test.txt

Notes:
See DOS Batch Files for other options.

CAUTION: This command DELETES the file(s) specified. Be sure to make backups of any files if needed.

If [file] is not found, the macro just continues. If you need to check for the existance of a file, use the *ExistFile* condition of the If-Then command.

Issue this command before issuing the WriteFile command if you want to ensure the WriteFile command is creating a new file, not just appending to an existing one.

The Delete command does not currently support long file names (at least entirely). However, you may use a ~? to represent the 7th character and beyond for all files with 9 or more characters in their prefix name (where ? is a number the operating system has assigned to represent the truncated portion of the name). For example to delete a file called *thisismyfile.tmp* from *c:\docs*, use Delete c:\docs\thisis~1.tmp. Note that one limitation is that *all* long file names (files with more than 8 characters at the prefix of their name) beginning with *thisis...* in the subdirectory *c:\docs* will be deleted. (Note: If there is more than one with a long name, the 7th position and beyond might be represented with a ~2, etc.). You may, however, use a wildcard to get all long file names. For example, to ensure all long files beginning with *thisis* are deleted, issue the command Delete c:\docs\thisis~?.*

If the total *File(s)* to be deleted are more than 1,000 or are on drive a or b, then a status window will display during the operation (since the operation is likely to take a couple seconds longer when either of those conditions are met).

Rename

Description:
Changes the name of either a file or directory.

Syntax:
RENAME OldName AS NewName

where *file* is the full path and file name for a file that you want to delete.

Example:
RENAME c:\tmp\test.txt AS c:\tmp\test.doc

Notes:
See DOS Batch Files for other options.

Remember to include the AS part of the command. (It is a required part of the command and it helps break up the command visually, especially when a file/directory has spaces in its name.)

The Name statement is similar to the operating system RENAME command, but it can also be used to change the name of an empty *directory* (an error occurs if the directory being renamed is not empty). For example, to rename the directory c:\12-25-97 to c:\12-26-97, use RENAME c:\12-25-97 AS c:\12-26-97. Also, Rename can move a file from one directory to another (on the same drive).

The arguments *OldName* and *NewName* should each contain a file name and an optional path (or just path if renaming an empty directory). If the path in NewName exists and is different from the path in OldName, the command moves the file to the new directory and renames the file if necessary. If NewName and OldName have different paths and the same file name, it moves the file to the new directory and leaves the file name unchanged.

If *OldName* is not found, an error occurs and you are given the option to Abort the macro, Retry the copy operation, or Ignore the error and continue with the rest of the macro. Use the *ExistFile* condition in the If-Then command if you want to avoid this error message entirely.

If *NewName* already exists as a file, an error occurs. Use the *ExistFile* condition in the If-Then command and/or Delete if you want to avoid this error.

Both NewName and OldName must be on the same drive.

Using Name on a file currently open produces an error. You must close an open file before renaming it.

MkDir

Description:
Like the DOS command, this will make a directory you specify.

Syntax:
MKDIR [path]

path is a string expression that identifies which directory you wish to make and should include the drive specification as part of the path. If you omit the drive, MkDir uses the current drive.

Example:
MKDIR c:\test\dir1

Notes:
See DOS Batch Files for other options.

MKDIR may be abbreviated with *MD*.

This command is also capable of creating several nested directories at once. For example, even if c:\test does not exist, if you issue MD c:\test\dir1, the command will create c:\test and then c:\dir1.

The MkDir command does not currently support long file names (at least entirely). If you specify a directory name of 9 or more characters, it will truncate it to the first 8 characters. For example, if you issue MkDir c:\thisismydirectory it will create a directory called *c:\thismydi*.

RmDir

Description:
Like the DOS command, this will remove a directory you specify.

Syntax:
RMDIR [path]

path is a string expression that identifies which directory you wish to remove and should include the drive specification as part of the path. If you omit the drive, RmDir will search the current drive for the directory.

Example:
RMDIR c:\test\dir1

Notes:
See DOS Batch Files for other options.

RMDIR may be abbreviated with *RD*.

The RmDir command does not currently support long file names (at least entirely). However, you may use a ~? to represent the 7th character and beyond for all directories with 9 or more characters in their name (where ? is a number the operating system has assigned to represent the truncated portion of the name). For example to remove a directory called *thisismydirectory*, use RmDir c:\thisis~1. (Note: If there is more than one with a long name, the 7th position and beyond might be represented with a ~2, ~3, etc.). Unlike the othe related commands (Copy, Delete, etc.), you may not use a wildcard to get all long directory names--you must know the numeric abbreviation for the directory.

Like the DOS RD command, you must delete/remove all files and subdirectories in the specified directory before you can remove it. If needed, issue the Delete command to delete files. To remove several nested directories, remove the deepest directory and move up. For example, if there is a directory called c:\test\dir1 and you want to remove it and all its files, use the following macro:

Deletes c:\test\dir1*.*
RD c:\test\dir1
Deletes c:\test*.*

RD c:\test

License Agreement

MACRO MANIA - PRODUCT LICENSE INFORMATION

NOTICE TO USERS: CAREFULLY READ THE FOLLOWING LEGAL AGREEMENT. USE OF THE SOFTWARE PROVIDED WITH THIS AGREEMENT (THE "SOFTWARE") CONSTITUTES YOUR ACCEPTANCE OF THESE TERMS. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT INSTALL AND/OR USE THIS SOFTWARE. USER'S USE OF THIS SOFTWARE IS CONDITIONED UPON COMPLIANCE BY USER WITH THE TERMS OF THIS AGREEMENT.

1. LICENSE GRANT. NorthStar Solutions grants you a license to use one copy of the version of this SOFTWARE on any one hardware product for as many licenses as you purchase. "You" means the company, entity or individual whose funds are used to pay the license fee. "Use" means storing, loading, installing, executing or displaying the SOFTWARE. You may not modify the SOFTWARE or disable any licensing or control features of the SOFTWARE except as an intended part of the SOFTWAREs programming features. When you first obtain a copy of the SOFTWARE, you are granted an evalution period of not more than 30 days, after which time you must pay for the SOFTWARE according to the terms and prices discussed in the SOFTWAREs documentation, or you must remove the SOFTWARE from your computer.

2. OWNERSHIP. The SOFTWARE is owned and copyrighted by NorthStar Solutions. Your license confers no title or ownership in the SOFTWARE and should not be construed as a sale of any right in the SOFTWARE .

3. COPYRIGHT. The SOFTWARE is protected by United States copyright law and international treaty provisions. You acknowledge that no title to the intellectual property in the SOFTWARE is transferred to you. You further acknowledge that title and full ownership rights to the SOFTWARE will remain the exclusive property of NorthStar Solutions and you will not acquire any rights to the SOFTWARE except as expressly set forth in this license. You agree that any copies of the SOFTWARE will contain the same proprietary notices which appear on and in the SOFTWARE.

4. REVERSE ENGINEERING. You agree that you will not attempt to reverse compile, modify, translate, or disassemble the SOFTWARE in whole or in part.

5. NO OTHER WARRANTIES. NORTHSTAR SOLUTIONS DOES NOT WARRANT THAT THE SOFTWARE IS ERROR FREE. NORTHSTAR SOLUTIONS DISCLAIMS ALL OTHER WARRANTIES WITH RESPECT TO THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES OR LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY MAY LAST, OR THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION.

6. SEVERABILITY. In the event of invalidity of any provision of this license, the parties agree that such invalidity shall not affect the validity of the remaining portions of this license.

7. NO LIABILITY FOR CONSEQUENTIAL DAMAGES. IN NO EVENT SHALL NORTHSTAR SOLUTIONS OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, SPECIAL, INCIDENTAL OR INDIRECT DAMAGES OF ANY KIND ARISING OUT OF THE DELIVERY, PERFORMANCE OR USE OF THE SOFTWARE, EVEN IF NORTHSTAR SOLUTIONS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL NORTHSTAR SOLUTIONS' LIABILITY FOR ANY CLAIM, WHETHER IN CONTRACT, TORT OR ANY OTHER

THEORY OF LIABILITY, EXCEED THE LICENSE FEE PAID BY YOU, IF ANY.

8. GOVERNING LAW. This license will be governed by the laws of the State of South Carolina as they are applied to agreements between South Carolina residents entered into and to be performed entirely within South Carolina . The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed.

9. ENTIRE AGREEMENT. This is the entire agreement between you and NorthStar Solutions which supersedes any prior agreement or understanding, whether written or oral, relating to the subject matter of this license.

Making Sure Keystrokes Get Received

There are generally three basic (and easily fixable) reasons a program might not be accepting keystrokes from a macro:

1) The program that should get the keystrokes has not been given the focus with either the Activate or Run command.

2) Step 1 listed above has been done, but then another Macro Mania command has been issued and has put the focus onto *Macro Mania* (and, thus, away from the program that should be getting the keystrokes). Since some of the Macro Mania commands put focus onto Macro Mania itself (e.g. Msg_Ok, GetInput, etc.), you should always be sure focus is on the program you want to get the keystrokes (and, thus, not on Macro Mania) by first using a Run or Activate command before using the SEND command.

3) The program getting the keystrokes needs time to catch up with Macro Mania....

Sometimes it is necessary to put in a short PAUSE command before using a SEND command to give the program receiving keystrokes time to get ready. Although most of the time adding a PAUSE is not necessary, if you experience that a program is not getting keystrokes (and you are sure you have addresses the two issues noted above), then it may be necessary to add a few second delay before using the SEND command. An example being worth a thousand words:

```
RUN myapp.exe
PAUSE 2
SEND {down 3}{home}+{end}%Ec
```

Although most applications are smart enough to start getting the keystrokes from the keyboard buffer after they launch, some seem to ignore the keyboard buffer and thus never receive what the macro has sent. On the same topic, it is often better to break up long SEND commands into several smaller ones. This is effective because Macro Mania will not begin processing the next line until the previous keystrokes have been sent and accepted by the receiving program. The following example shows what I mean:

```
SEND abcdefg^G{tab}{up 4}qrstuv
```

is changed to

```
SEND abcdefg
SEND ^G
SEND {tab}{up 4}
SEND qrstuv
```

As shown above, this also tends to make macros easier to read and follow.