

\$₁ #₂ +₃ K₄ **Macro Language**

The MicroEMACS macro language allows you to add extensions to the editor. Statements (one per line) are composed of the following elements:

Commands manipulate text, buffers, windows, etc... within the editor

Directives control the flow of execution within a macro

Arguments:

Constants

Variables

Functions

Comments

Macros are registered with MicroEMACS by the store-macro or store-procedure commands. They get executed through menus or keystrokes they have been bound to, or through the execute-macro-*n* or run commands.

Macros can also be executed directly from a buffer or a file by the execute-buffer or execute-file commands.

1\$ Macro Language

2# MacroLanguage

3+ Index:2040

4K macro;language

\$₅ #₆ +₇ K₈ **Commands**

By topic:

Binding

Block of Text

Buffer, Window and Screen

Clipboard and Kill Buffer

Execution, Macro and Variable

File

Mouse

Positioning

Search and Replace

Miscellaneous

Alphabetical lists:

Standard commands

Additional commands

5\$ Commands

6# Commands

7+ MacroLanguage:010

8^K commands

\$₉ #₁₀ +₁₁ K₁₂ **Binding commands**

[apropos](#)
[bind-to-key](#)
[bind-to-menu](#)
[ctx-prefix](#)
[describe-bindings](#)
[describe-key](#)
[macro-to-key](#)
[macro-to-menu](#)
[meta-prefix](#)
[unbind-key](#)
[unbind-menu](#)

9\$ Binding commands
10# BindingCommands
11+ CommandsByTopic:bindingcommands
12^K binding;commands

\$₁₃ #₁₄ +₁₅ K₁₆ **Block of Text commands**

Commands that affect regions, lines, words and paragraphs.

case-region-lower
case-region-upper
case-word-capitalize
case-word-lower
case-word-upper
copy-region
count-words
delete-blank-lines
delete-next-word
delete-previous-word
deta-region
enta-region
fill-paragraph
indent-region
kill-paragraph
kill-region
kill-to-end-of-line
narrow-to-region
remove-mark
set-fill-column
set-mark
trim-region
undent-region
widen-from-region
wrap-word

13\$ Block of Text commands

14# BlockOfTextCommands

15+ CommandsByTopic:blockoftextcommands

16K region;line;word;paragraph;commands

\$₁₇ #₁₈ +₁₉ K₂₀ **Buffer, Window and Screen commands**

add-global-mode
add-mode
cascade-screens
change-screen-column
change-screen-row
change-screen-size
change-screen-width
clear-and-redraw
cycle-screens
delete-buffer
delete-global-mode
delete-other-windows
delete-mode
delete-screen
delete-window
execute-buffer
filter-buffer
find-screen
grow-window
list-buffers
list-screens
maximize-screen
minimize-screen
move-window-down
move-window-up
name-buffer
narrow-to-region
next-buffer
next-window
pipe-command
pop-buffer
previous-window
rename-screen
resize-window
restore-screen
restore-window
save-window
scroll-next-up
scroll-next-down
select-buffer
shrink-window
split-current-window
tile-screens
unmark-buffer
update-screen
widen-from-region

17\$ Buffer, Window and Screen commands

18# BufferWindowScreenCommands

19+ CommandsByTopic:bufferwindowsscreencommands

20K buffer;window;screen;commands

\$₂₁ #₂₂ +₂₃ K₂₄ **Clipboard and Kill Buffer commands**

clip-region

copy-region

cut-region

cycle-ring

delete-kill-ring

delete-next-character (with argument)

delete-next-word

delete-previous-character (with argument)

delete-previous-word

insert-clip

kill-paragraph

kill-region

kill-to-end-of-line

yank

yank-pop

21\$ Clipboard and Kill Buffer commands

22# ClipboardKillBufferCommands

23+ CommandsByTopic:clipboardandkillbuffercommands

24K clipboard;kill;commands

\$₂₅ #₂₆ +₂₇ K₂₈ **Execution, Macro and Variable commands**

abort-command
begin-macro
describe-functions
describe-variables
display
end-macro
execute-buffer
execute-command-line
execute-file
execute-macro
execute-macro-*n*
execute-named-command
execute-procedure
execute-program
filter-buffer
i-shell
nop
pipe-command
run
set
shell-command
source
store-macro
store-procedure
help-engine

25\$ Execution, Macro and Variable commands

26# ExecutionMacroVariableCommands

27+ CommandsByTopic:executionmacrovariablecommands

28K execute;spawn;filter;pipe;shell;DOS;macro;variable;commands

\$₂₉ #₃₀ +₃₁ K₃₂ **File Commands**

append-file
change-file-name
execute-file
find-file
insert-file
read-file
save-file
show-files
source
view-file
write-file

29\$ File Commands

30# FileCommands

31+ CommandsByTopic:filecommands

32K file;commands

\$₃₃ #₃₄ +₃₅ K₃₆ **Mouse commands**

mouse-move

mouse-move-down

mouse-move-up

mouse-region-down

mouse-region-up

mouse-resize-screen

33\$ Mouse commands

34# MouseCommands

35+ CommandsByTopic:mousecommands

36K mouse;commands

\$³⁷ #³⁸ +³⁹ K⁴⁰ **Positioning commands**

backward-character
beginning-of-file
beginning-of-line
buffer-position
end-of-file
end-of-line
end-of-word
exchange-point-and-mark
forward-character
goto-line
goto-mark
goto-matching-fence
next-line
next-page
next-paragraph
next-word
previous-line
previous-page
previous-paragraph
previous-word
redraw-display

37\$ Positioning commands

38# PositioningCommands

39+ CommandsByTopic:positioningcommands

40K position;point;commands

\$₄₁ #₄₂ +₄₃ K₄₄ **Search and Replace commands**

hunt-backward

hunt-forward

incremental-search

query-replace-string

replace-string

reverse-incremental-search

search-forward

search-reverse

41\$ Search and Replace commands

42# SearchReplaceCommands

43+ CommandsByTopic:searchreplacecommands

44K search;replace;commands

\$₄₅ #₄₆ +₄₇ K₄₈ **Miscellaneous Commands**

clear-message-line
exit-emacs
handle-tab
help
insert-space
insert-string
newline
newline-and-indent
nop
open-line
overwrite-string
print
quick-exit
quote-character
redraw-display
set-encryption-key
set-fill-column
transpose-characters
universal-argument
write-message

45\$ Miscellaneous commands
46# MiscellaneousCommands
47+ CommandsByTopic:zzz010
48K misc;commands

\$49 #50 +51 K52 **Standard commands**

The following commands are available in all implementations of MicroEMACS:

<u>abort-command</u>	Allows the user to abort out of any command that is waiting for input
<u>add-global-mode</u>	Add a global mode for all new <u>buffers</u>
<u>add-mode</u>	Add a mode to the current <u>buffer</u>
<u>append-file</u>	Append a <u>buffer</u> to the end of a file
<u>apropos</u>	Lists <u>commands</u> and <u>macros</u> whose name contains the string specified
<u>backward-character</u>	Move one character to the left
<u>begin-macro</u>	Begin recording a <u>keyboard macro</u>
<u>beginning-of-file</u>	Move to the beginning of the file in the current <u>buffer</u>
<u>beginning-of-line</u>	Move to the beginning of the current line
<u>bind-to-key</u>	<u>Bind</u> a key to a <u>command</u>
<u>buffer-position</u>	List the position of the <u>point</u> on the <u>message line</u>
<u>case-region-lower</u>	Make a <u>region</u> all lower case
<u>case-region-upper</u>	Make a <u>region</u> all upper case
<u>case-word-capitalize</u>	Capitalize the following word
<u>case-word-lower</u>	Lower case the following word
<u>case-word-upper</u>	Upper case the following word
<u>change-file-name</u>	Change the name of the file in the current <u>buffer</u>
<u>change-screen-column</u>	change the column offset of the current <u>screen</u>
<u>change-screen-row</u>	change the row offset of the current <u>screen</u>
<u>change-screen-size</u>	Change the number of lines of the current <u>screen</u>
<u>change-screen-width</u>	Change the number of columns of the current <u>screen</u>
<u>clear-and-redraw</u>	Repaint all <u>screens</u> or center the <u>point</u> in the current <u>window</u>
<u>clear-message-line</u>	Clear the <u>message line</u>
<u>copy-region</u>	Copy the current <u>region</u> into the <u>kill buffer</u>
<u>count-words</u>	Count how many words, lines and characters are in the current <u>region</u>
<u>ctlx-prefix</u>	<u>Bound</u> to the key used as the ^X prefix
<u>cycle-ring</u>	moves the current position of the <u>kill buffer</u> within the <u>kill ring</u>
<u>cycle-screens</u>	Bring the rearmost <u>screen</u> to front
<u>delete-blank-lines</u>	Delete all blank lines around the <u>point</u>

49\$ Standard commands
50# StandardCommands
51+ CommandsByTopic:zzz900
52K standard;commands

<u>delete-buffer</u>	Delete a <u>buffer</u> which is not being currently displayed in a <u>window</u>
<u>delete-kill-ring</u>	Reclaim the memory used by the <u>kill ring</u>
<u>delete-global-mode</u>	Turn off a global mode
<u>delete-mode</u>	Turn off a mode in the current <u>buffer</u>
<u>delete-next-character</u>	Delete the character following the <u>point</u>
<u>delete-next-word</u>	Delete the word following the <u>point</u>
<u>delete-other-windows</u>	Make the current <u>window</u> cover the entire <u>screen</u>
<u>delete-previous-character</u>	Delete the character to the left of the <u>point</u>
<u>delete-previous-word</u>	Delete the word to the left of the <u>point</u>
<u>delete-screen</u>	Delete a <u>screen</u> (not the top one)
<u>delete-window</u>	Remove the current <u>window</u> from the <u>screen</u>
<u>describe-bindings</u>	List all <u>commands</u> and <u>macros</u>
<u>describe-functions</u>	List all <u>functions</u>
<u>describe-variables</u>	List all <u>variables</u>
<u>describe-key</u>	Describe what <u>command</u> or <u>macro</u> is <u>bound</u> to a <u>keystroke</u> sequence
<u>detab-region</u>	Change all tabs in a <u>region</u> to the equivalent spaces
<u>display</u>	Displays a <u>variable's</u> current value
<u>end-macro</u>	Stop recording a <u>keyboard macro</u>
<u>end-of-file</u>	Move to the end of the current <u>buffer</u>
<u>end-of-line</u>	Move to the end of the current line
<u>end-of-word</u>	Move just past the end of the current word
<u>entab-region</u>	Change multiple spaces to tabs where possible
<u>exchange-point-and-mark</u>	Move the <u>point</u> to the last <u>marked</u> spot, make the original position be marked
<u>execute-buffer</u>	Execute a <u>buffer</u> as a <u>macro</u>
<u>execute-command-line</u>	Execute a line typed on the <u>command line</u> as a <u>macro</u>
<u>execute-file</u>	Execute a file as a <u>macro</u>
<u>execute-macro</u>	Execute the <u>keyboard macro</u> (play back the recorded keystrokes)
<u>execute-macro-n</u>	Execute numbered <u>macro</u> <i>n</i> where <i>n</i> is an integer from 1 to 40
<u>execute-named-command</u>	Execute a <u>command</u> by name
<u>execute-procedure</u>	Execute a <u>procedure</u> by name
<u>execute-program</u>	Execute a program directly (not through an intervening shell)
<u>exit-emacs</u>	Exit MicroEMACS. If there are unwritten, changed <u>buffers</u> MicroEMACS will ask to confirm
<u>fill-paragraph</u>	Fill the current paragraph
<u>filter-buffer</u>	Filter the current <u>buffer</u> through an external filter

<u>find-file</u>	Find a file to edit in the current <u>window</u>
<u>find-screen</u>	Bring the named <u>screen</u> on top, creating it if needed
<u>forward-character</u>	Move one character to the right
<u>goto-line</u>	Goto a numbered line
<u>goto-mark</u>	Goto a numbered <u>mark</u>
<u>goto-matching-fence</u>	Goto the matching fence
<u>grow-window</u>	Make the current <u>window</u> larger
<u>handle-tab</u>	Insert a tab or set tab stops
<u>hunt-backward</u>	Hunt for the last match of the last search string
<u>hunt-forward</u>	Hunt for the next match of the last search string
<u>help</u>	Read EMACS.HLP into a <u>buffer</u> and display it
<u>i-shell</u>	Shell up to a new command processor
<u>incremental-search</u>	Search for a string, incrementally
<u>indent-region</u>	Indent the current <u>region</u> one tab
<u>insert-file</u>	Insert a file at the <u>point</u> in the current file
<u>insert-space</u>	Insert a space to the right of the <u>point</u>
<u>insert-string</u>	Insert a string at the <u>point</u>
<u>kill-paragraph</u>	Delete the current paragraph
<u>kill-region</u>	Delete the current <u>region</u> , moving it to the <u>kill buffer</u>
<u>kill-to-end-of-line</u>	Delete the rest of the current line
<u>list-buffers</u>	List all existing <u>buffers</u>
<u>list-screens</u>	List all existing <u>screens</u>
<u>macro-to-key</u>	<u>Bind</u> a key to a <u>macro</u>
<u>meta-prefix</u>	Key used to precede all <u>META</u> commands
<u>mouse-move</u>	Usually bound to the movement of the mouse
<u>mouse-move-down</u>	Usually bound to a press on the left mouse button
<u>mouse-move-up</u>	Usually bound to the release of the left mouse button
<u>mouse-region-down</u>	Usually bound to a press on the right mouse button
<u>mouse-region-up</u>	Usually bound to the release of the right mouse button
<u>mouse-resize-screen</u>	Resize the screen to bring the bottom-left corner where the mouse was clicked
<u>move-window-down</u>	Scroll the current <u>window</u> down
<u>move-window-up</u>	Scroll the current <u>window</u> up
<u>name-buffer</u>	Change the name of the current <u>buffer</u>
<u>narrow-to-region</u>	Hides all text not in the current <u>region</u> (see <u>widen-from-region</u>)
<u>newline</u>	Insert a newline
<u>newline-and-indent</u>	Insert a newline and indent the new line the same as the preceding line

<u>next-buffer</u>	Bring the next <u>buffer</u> in the list into the current <u>window</u>
<u>next-line</u>	Move down one line
<u>next-page</u>	Move down one page
<u>next-paragraph</u>	Move to the next paragraph
<u>next-window</u>	Move to the next <u>window</u>
<u>next-word</u>	Move to the beginning of the next word
<u>nop</u>	Does nothing
<u>open-line</u>	Open a line at the <u>point</u>
<u>overwrite-string</u>	Overwrite a string at the <u>point</u>
<u>pipe-command</u>	Execute an external command and place its output in a <u>buffer</u>
<u>pop-buffer</u>	Display a <u>buffer</u> temporarily, paging through it
<u>previous-line</u>	Move up one line
<u>previous-page</u>	Move up one page
<u>previous-paragraph</u>	Move back one paragraph
<u>previous-window</u>	Move to the last <u>window</u>
<u>previous-word</u>	Move to the beginning of the word to the left of the <u>point</u>
<u>print</u>	Display a string on the <u>message line</u> (synonym of <u>write-message</u>)
<u>query-replace-string</u>	Replace occurrences of a string with another string, interactively querying the user
<u>quick-exit</u>	Exit MicroEMACS, writing out all the changed <u>buffers</u>
<u>quote-character</u>	Insert the next character literally
<u>read-file</u>	Read a file into the current <u>buffer</u>
<u>redraw-display</u>	Reposition the current line in the <u>window</u>
<u>remove-mark</u>	Remove a numbered <u>mark</u>
<u>replace-string</u>	Replace all occurrences of a string with another string
<u>resize-window</u>	Change the number of lines in the current <u>window</u>
<u>restore-window</u>	Move to the last saved <u>window</u> (see <u>save-window</u>)
<u>reverse-incremental-search</u>	Search backwards, incrementally
<u>run</u>	Execute a named <u>procedure</u>
<u>save-file</u>	Save the current <u>buffer</u> if it is changed
<u>save-window</u>	Remember the current <u>window</u> (see <u>restore-window</u>)
<u>scroll-next-up</u>	Scroll the next <u>window</u> up
<u>scroll-next-down</u>	Scroll the next <u>window</u> down
<u>search-forward</u>	Search for a string
<u>search-reverse</u>	Search backwards for a string
<u>select-buffer</u>	Select a <u>buffer</u> to display in the current <u>window</u>
<u>set</u>	Set a <u>variable</u> to a value

<u>set-encryption-key</u>	Set the encryption key of the current <u>buffer</u>
<u>set-fill-column</u>	Set the current fill column
<u>set-mark</u>	Set a numbered <u>mark</u>
<u>shell-command</u>	Causes an external shell to execute a command
<u>show-files</u>	list files matching a pattern within a directory
<u>shrink-window</u>	Make the current <u>window</u> smaller
<u>source</u>	Execute a file as a <u>macro</u>
<u>split-current-window</u>	Split the current <u>window</u> in two
<u>store-macro</u>	Store the following <u>macro</u> lines as a numbered macro
<u>store-procedure</u>	Store the following <u>macro</u> lines in a named procedure
<u>transpose-characters</u>	Transpose the character at the <u>point</u> with the character immediately to the left
<u>trim-region</u>	Trim any trailing white space from a <u>region</u>
<u>unbind-key</u>	<u>Unbind</u> a key from a <u>command</u> or <u>macro</u>
<u>undent-region</u>	Remove a leading indent from a <u>region</u>
<u>universal-argument</u>	Execute the following <u>command</u> or <u>macro</u> 4 times
<u>unmark-buffer</u>	Unmark the current <u>buffer</u> (so it is no longer seen as changed)
<u>update-screen</u>	Force a display update during <u>macro</u> execution
<u>view-file</u>	Read a file in a <u>buffer</u> , in view mode
<u>widen-from-region</u>	Restores hidden text (see <u>narrow-to-region</u>)
<u>wrap-word</u>	Wrap the current word (internal command)
<u>write-file</u>	Write the current <u>buffer</u> under a new file name
<u>write-message</u>	Display a string on the <u>message line</u>
<u>yank</u>	Yank the <u>kill buffer</u> into the current <u>buffer</u> at the <u>point</u>
<u>yank-pop</u>	yank the <u>kill buffer</u> , subsequent invocations replacing the yanked text by the next one from the <u>kill ring</u> .

\$₅₃ #₅₄ +₅₅ K₅₆ **Additional commands**

The following commands are available only from the Microsoft Windows version of MicroEMACS:

<u>bind-to-menu</u>	creates a menu item and <u>binds</u> it to a <u>command</u>
<u>cascade-screens</u>	arranges all non-iconic <u>screens</u> using a cascading scheme
<u>clip-region</u>	copies the <u>region</u> to the Windows <u>clipboard</u>
<u>cut-region</u>	moves the <u>region</u> to the Windows <u>clipboard</u>
<u>help-engine</u>	invokes the Microsoft Windows help engine
<u>insert-clip</u>	inserts the contents of the Windows <u>clipboard</u> at the <u>point</u>
<u>macro-to-menu</u>	creates a menu item and binds it to a <u>macro</u>
<u>maximize-screen</u>	makes the current <u>screen</u> occupy the whole MicroEMACS window
<u>minimize-screen</u>	iconizes the current <u>screen</u>
<u>rename-screen</u>	change the current <u>screen</u> 's name
<u>restore-screen</u>	restores the current <u>screen</u> back from maximized or iconized state
<u>tile-screens</u>	arranges all non-iconic <u>screens</u> using a tiling scheme
<u>unbind-menu</u>	deletes a menu item

53\$ Additional commands

54# AdditionalCommands

55+ CommandsByTopic:zzz910

56K Additionnal commands;Microsoft Windows;MS Windows

\$₅₇ #₅₈ +₅₉ K₆₀ **Directives**

Directives are used within macros to control what lines are executed and in what order.

Directives always start with the exclamation mark "!" character and must be the first non-white text placed on a line. They are:

!BREAK
!ENDM
!FORCE
!GOTO
!IF, !ELSE and !ENDIF
!RETURN
!WHILE and !ENDWHILE

Directives do not make sense as a single commands. As such, they cannot be called up singly or bound to keystrokes. Directives executed interactively (via the execute-command-line command) are ignored.

57\$ Directives

58# Directives

59+ MacroLanguage:020

60K directives;macro

\$₆₁ #₆₂ +₆₃ K₆₄ **!BREAK**

This directive lets you abort out of the most inner currently executing while loop, in a macro. It is often used to abort processing for error conditions. For example:

```
; Read in files and substitute "beginning" with "beginning"
set %filename #list
!while &not &seq %filename "<end>"
!force      find-file %filename
            !if &seq $status FALSE
              write-message "[File read error]"
              !break
            !endif
            beginning-of-file
            replace-string "beginning" "beginning"
            save-file
            set %filename #list
!endwhile
```

61\$!BREAK directive

62# .BREAK

63+ Directives: BREAK

64K !BREAK;!WHILE;loop

\$⁶⁵ #⁶⁶ +⁶⁷ K⁶⁸ !ENDM

This directive is used to terminate a macro being stored. For example:

```
; Read in a file in view mode, and make the window red
store-procedure get-red-viewed-file
  view-file @"File to view: "
  add-mode "red"
!endm
```

Related commands:

store-procedure
store-macro.

65\$!ENDM directive

66# .ENDM

67+ Directives:endm

68K !ENDM;macro

\$69 #70 +71 K72 **!FORCE**

When MicroEMACS executes a macro, if any command fails, the macro is terminated at that point. If a line is preceded by a **!FORCE** directive, execution continues whether the command succeeds or not.

This is often used together with the \$status variable to test if a command succeeded. For example:

```
set %seekstring @"String to Find: "  
!force search-forward %seekstring  
!if $status  
    print "Your string is Found"  
!else  
    print "No such string!"  
!endif
```

69\$!FORCE directive
70# .FORCE
71+ Directives:force
72K !FORCE;error

\$₇₃ #₇₄ +₇₅ K₇₆ !GOTO

The flow of execution within a MicroEMACS macro can be controlled using the !GOTO directive. It takes a label as argument. A label consists of a line starting with an asterisk "*" and then an alphanumeric label. Only labels in the currently executing macro can be jumped to, and trying to jump to a non-existing label terminates execution of a macro. For example:

```
; Create a block of DATA statements for a BASIC program
insert-string "1000 DATA "
set %linenum 1000
*nxtin
update-screen      ;make sure we see the changes
set %data @@ "Next number: "
!if &equal %data 0
    !goto finish
!endif
!if &greater $curcol 60
    2 delete-previous-character
    newline
    set %linenum &add %linenum 10
    insert-string &cat %linenum " DATA "
!endif
insert-string &cat %data ", "
!goto nxtin
*finish
2 delete-previous-character
newline
```

Note that loops constructed with !WHILE are usually more efficient than those constructed purely by !GOTOS.

73\$!GOTO directive
74# .GOTO
75+ Directives:goto
76K !GOTO;!WHILE;loop

\$77 #78 +79 K80 **!IF, !ELSE and !ENDIF**

The **!IF** directive allows for conditional execution within a macro.

Lines following the **!IF** directive, until the corresponding **!ELSE** or **!ENDIF**, are executed only if the expression within the **!IF** line evaluates to a TRUE value. Lines following an **!ELSE** directive, until the corresponding **!ENDIF**, are executed only if the expression within the corresponding **!IF** line did not evaluate to a TRUE value.

For example, the following macro creates the portion of a text file automatically:

```
!if &sequal %curplace "timespace vortex"
    insert-string "First, rematerialize~n"
!endif
!if &sequal %planet "earth"          ;If we have landed on earth...
    !if &sequal %time "late 20th century"    ;and we are then
        write-message "Contact U.N.I.T."
    !else
        insert-string "Investigate the situation....~n"
        insert-string "(SAY 'stay here Sarah)~n"
    !endif
!else
    set %conditions @"Atmosphere conditions outside? "
    !if &sequal %conditions "safe"
        insert-string &cat "Go outside....." "~n"
        insert-string "lock the door~n"
    !else
        insert-string "Dematerialize..try somewhen else"
        newline
    !endif
!endif
```

77\$ **!IF, !ELSE and !ENDIF** directives

78# **.IF**

79+ Directives:**if**

80K **!IF;!ELSE;!ENDIF**

`$81 #82 +83 K84 !RETURN`

This directive causes the current macro to exit, either returning to the caller (if any) or to interactive mode. For example:

```
; Check the display type and set %wintyp
!if &sequal $sres "MSWIN"
    set %wintyp 1
    !return
!endif
set %wintyp 0
write-message "You are not running under MS-Windows!"
!return
```

81\$!RETURN directive

82# .RETURN

83+ Directives:return

84K !RETURN

\$⁸⁵ #⁸⁶ +⁸⁷ K⁸⁸ **!WHILE and !ENDWHILE**

This pair of directives facilitates repetitive execution within a macro. If a group of statements needs to be executed while a certain expression evaluates to TRUE, enclose them with a while loop. For example:

```
!while &less $curcol 70
    insert-string &cat &cat "[" #stuff "]"
!endwhile
```

While loops may be nested and can contain and be the targets of !GOTOs with no ill effects. Using a while loop to enclose a repeated task will run much faster than the corresponding construct using !IFs.

85\$!WHILE and !ENDWHILE directives

86# .WHILE

87+ Directives:while

88K !WHILE;!ENDWHILE;loop;!BREAK

\$⁸⁹ #⁹⁰ +⁹¹ K⁹² Arguments

In the MicroEMACS macro language, commands and functions often require arguments. There are three types of arguments and they are automatically converted to the proper type when used:

Numerical	An ASCII string of digits which is interpreted as a numeric value. Any string which does not start with a digit or a minus sign "-" will be considered zero.
String	An arbitrary string of characters. Strings are limited to 128 characters in length.
Boolean	A logical value consisting of the string "TRUE" or "FALSE". Numeric strings will also evaluate to "FALSE" if they are equal to zero, and "TRUE" if they are non-zero. Arbitrary text strings will be considered equivalent "FALSE".

While arguments usually follow the command or function that uses them, a single numerical argument can also be placed in front of a command, producing an effect similar to the numeric arguments used in interactive mode.

If a command needs more arguments than have been supplied on the line, the command fails.

89\$ Arguments

90# Arguments

91+ MacroLanguage:030

92K argument;evaluate;expression;

Wherever macro language statements need to have arguments, it is legal to place constants. A constant is a double quote character, followed by a string of characters, and terminated by another double quote character.

The double quotes around constants are not needed if the constant contains no white space and it also does not happen to meet the rules for any other MicroEMACS commands, directives, variables, or functions. This is very practical for numeric constants.

To represent various special characters within a constant, the tilde "~" character is used. The character following the tilde is interpreted according to the following table:

Sequence	Meaning
~"	double quote
~~	tilde
~b	backspace (^H)
~f	formfeed (^L)
~l	linefeed (^J)
~n	newline
~r	carriage return (^M)
~t	tab (^I)

Any character not in the above table which follows a tilde will be passed unmodified. This action is similar to the quote-character (^Q) command available from the keyboard.

MicroEMACS may use different characters for line terminators on different computers. The "~n" combination will always get the proper line terminating sequence for the current system.

\$⁹⁷ #⁹⁸ +⁹⁹ K¹⁰⁰ Variables

Variables are part of the MicroEMACS Macro language. They can be used wherever an argument (number, boolean or string) is needed.

Environmental variables both control and report on different aspects of the editor. User variables hold values which may be changed and inspected. Buffer variables allow lines from buffers to be used as values. Interactive variables allow macros to prompt the user for information.

97\$ Variables

98# Variables

99+ MacroLanguage:050

100K variable;macro

\$₁₀₁ #₁₀₂ +₁₀₃ K₁₀₄ **Buffer Variables**

Buffer variables are a way to take a line of text from a buffer and place it in a variable. They can only be queried and cannot be set. A buffer variable consists of the buffer name, preceded by a pound sign "#". Its value is the text between the point and the end of the line. Each use of a buffer variable advances the point to the beginning of the following line.

For example, if you have a buffer by the name of RIGEL2, and it contains the text (the point being on the "B" of "Bloomington"):

```
Richmond
Lafayette
Bloomington
Indianapolis
Gary
```

and within a command you reference #rigel2, like in:

```
insert-string #rigel2
```

MicroEMACS would start at the current point in the RIGEL2 buffer and grab all the text up to the end of that line and pass that back. Then it would advance the point to the beginning of the next line. Thus, after the insert-string command executes, the string "Bloomington" gets inserted into the current buffer, and the buffer RIGEL2 now looks like this (the point is on the "I" of "Indianapolis"):

```
Richmond
Lafayette
Bloomington
Indianapolis
Gary
```

When the end of a buffer variable is reached, the value returned is: <END>

101\$ Buffer Variables
102# BufferVariables
103+ Variables:buffervariables
104K buffer;variable

\$₁₀₅ #₁₀₆ +₁₀₇ K₁₀₈ **Environmental Variables**

These variables are used to change or get information about various aspects of the editor. They return a current setting if used as part of an expression. All environmental variable names begin with a dollar sign "\$" and are in lower case:

<u>\$acount</u>	Countdown until next auto-save
<u>\$asave</u>	Auto-save frequency
<u>\$bufhook</u>	Command/macro run when entering a buffer
<u>\$cbflags</u>	Buffer attribute flags.
<u>\$cbufname</u>	Buffer name
<u>\$cfname</u>	File name
<u>\$cmdhook</u>	Command/macro run before each keystroke
<u>\$cmode</u>	Buffer modes
<u>\$cuchar</u>	ASCII value of character
<u>\$curcol</u>	Current column
<u>\$curline</u>	Current line
<u>\$curwidth</u>	Number of columns
<u>\$curwind</u>	Window index
<u>\$cwinline</u>	Line number in current window
<u>\$debug</u>	Macro debugging flag
<u>\$deskcolor</u>	Color for desktop
<u>\$diagflag</u>	Diagonal dragging flag
<u>\$discmd</u>	Prompt echo flag
<u>\$disinp</u>	Input echo flag
<u>\$disphigh</u>	High-bit characters display flag
<u>\$exbhook</u>	Command/macro run when leaving a buffer.
<u>\$fcol</u>	Line number at top of window
<u>\$fillcol</u>	Fill column.
<u>\$flicker</u>	Flicker flag (for CGA or animated grinder cursor)
<u>\$fmtlead</u>	Text formatter command prefixes
<u>\$gflags</u>	Global flags
<u>\$gmode</u>	Global mode flags
<u>\$hardtab</u>	Size of hard tabs
<u>\$hlight</u>	Region to be highlighted
<u>\$hjump</u>	Horizontal scrolling quantum
<u>\$hscroll</u>	Horizontal scrolling flag
<u>\$hscribar</u>	Horizontal scroll bar flag
<u>\$iserm</u>	Incremental search string terminator key
<u>\$kill</u>	Kill buffer contents
<u>\$language</u>	National language used by MicroEMACS
<u>\$lastkey</u>	Last keyboard character
<u>\$lastmesg</u>	Last message
<u>\$line</u>	Current line contents
<u>\$lterm</u>	Line terminator string
<u>\$lwidth</u>	Width of current line
<u>\$match</u>	Last string matched in a search
<u>\$mmove</u>	Controls the generation of mouse movements
<u>\$modeflag</u>	Mode line display flag
<u>\$msflag</u>	Mouse flag

105\$ Environmental Variables

106# EnvironmentalVariables

107+ Variables:environmentalvariables

108K variable

<u>\$numwind</u>	Number of windows
<u>\$oldcrypt</u>	Encryption method flag
<u>\$orgrow</u>	Row of current screen within desktop
<u>\$orgcol</u>	Column of current screen within desktop
<u>\$os</u>	Operating system (MSWIN under MS-Windows)
<u>\$overlap</u>	Size of overlap during paging
<u>\$pagelen</u>	Number of lines in screen
<u>\$palette</u>	Color palette settings
<u>\$paralead</u>	Paragraph start characters
<u>\$pending</u>	Keystrokes pending flag
<u>\$popflag</u>	Popup buffer flag
<u>\$posflag</u>	Row&column display flag
<u>\$progname</u>	"MicroEMACS"
<u>\$readhook</u>	Command/macro run when a file is read
<u>\$region</u>	Contents of current region
<u>\$replace</u>	Default replace string.
<u>\$rval</u>	Exit value from last invoked subprocess
<u>\$scrname</u>	Screen name
<u>\$search</u>	Default search string
<u>\$searchpnt</u>	After-search-positioning flag
<u>\$seed</u>	Random number generator seed
<u>\$softtab</u>	Tab size for handle-tab command
<u>\$sres</u>	Display resolution (MSWIN under MS-Windows)
<u>\$ssave</u>	Safe-save flag
<u>\$sscroll</u>	Smooth scroll flag
<u>\$status</u>	Status from last command
<u>\$sterm</u>	Search string terminator key
<u>\$target</u>	Target for line moves
<u>\$time</u>	Date and time
<u>\$timeflag</u>	Time display flag
<u>\$tpause</u>	Duration of fence matching pause
<u>\$version</u>	MicroEMACS version
<u>\$vscrlbar</u>	Vertical scroll bar flag
<u>\$wchars</u>	List of characters that can be part of a word
<u>\$wline</u>	Window height (lines)
<u>\$wraphook</u>	Command/macro run when wrapping text
<u>\$writehook</u>	Command/macro run when writing a file
<u>\$xpos</u>	Column the mouse was in at last click
<u>\$yankflag</u>	After-yank-positioning flag
<u>\$ypos</u>	Line the mouse was in at last click

\$₁₀₉ #₁₁₀ +₁₁₁ K₁₁₂ \$account

This variable is used in ASAVE mode. It contains the countdown on inserted character until the next auto-save. When it reaches zero, it is reset to the value of \$asave.

Initial value: 256

109\$ \$account variable
110# _account
111+ EVariables:account
112K \$account;ASAVE;autosave

\$₁₁₃ #₁₁₄ +₁₁₅ K₁₁₆ **\$asave**

This variable is used in ASAVE mode. It specifies the value used to reset \$acount after an automatic save occurs.

Default value: 256

113\$ \$asave variable

114# _asave

115+ EVariables:asave

116K \$asave;ASAVE;autosave

\$₁₁₇ #₁₁₈ +₁₁₉ K₁₂₀ **\$bufhook**

The command or macro named in this variable is run when a buffer is entered. This can be used to implement modes which are specific to a particular file or file type.

Default value: nop

117\$ \$bufhook variable
118# _bufhook
119+ EVariables:bufhook
120K \$bufhook;hook;buffer

\$₁₂₁ #₁₂₂ +₁₂₃ K₁₂₄ **\$cbflags**

This variable contains the current buffer's attribute flags, encoded as the sum of the following numbers:

- 1 Internal invisible buffer
- 2 Changed since last read or write
- 4 Buffer was truncated when read (due to lack of memory)
- 8 Buffer has been narrowed

Only the invisible (1) and changed (2) flags can be modified by setting \$cbflags. The truncated file (4) and narrowed (8) flags are read-only.

121\$ \$cbflags variable
122# _cbflags
123+ EVariables:cbflags
124K \$cbflags

$\$_{125} \#_{126} +_{127} K_{128} \textbf{\$cbufname}$

This variable contains the name of the current buffer.

125\$ \$cbufname variable
126# _cbufname
127+ EVariables:cbufname
128K \$cbufname;buffer

$\$_{129} \#_{130} +_{131} K_{132} \textbf{\$cfname}$

This variable contains the file name associated to the current buffer.

129\$ \$cfname variable
130# _cfname
131+ EVariables:cfname
132K \$cfname;file;buffer

\$₁₃₃ #₁₃₄ +₁₃₅ K₁₃₆ \$cmdhook

This variable contains the name of a command or macro to run before accepting a keystroke.
This is by default set to the nop command.

Default value: nop

133\$ \$cmdhook variable

134# _cmdhook

135+ EVariables:cmdhook

136K \$cmdhook;hook;keyboard

\$137 #138 +139 K140 **\$cmode and \$gmode**

The two variables \$cmode and \$gmode contain a number that corresponds to the modes for the current buffer (\$cmode) and the new buffers (\$gmode). They are encoded as the sum of the following numbers for each of the possible modes:

<u>WRAP</u>	1	Word wrap
<u>CMODE</u>	2	C indentation and fence matching
<u>SPELL</u>	4	Interactive spell checking (Not implemented yet)
<u>EXACT</u>	8	Exact matching for searches
<u>VIEW</u>	16	Read-only buffer
<u>OVER</u>	32	Overwrite mode
<u>MAGIC</u>	64	Regular expressions in search
<u>CRYPT</u>	128	Encryption mode active
<u>ASAVE</u>	256	Auto-save mode

Thus, if you wished to set the current buffer to have CMODE, EXACT, and MAGIC on, and all the others off, you would add up the values for those three, CMODE 2 + EXACT 8 + MAGIC 64 = 74, and use a statement like:

```
set $cmode 74
```

or, use the binary or operator to combine the different modes:

```
set $cmode &bor &bor 2 8 64
```

Alternatively, you can also modify the modes one by one, using the add-mode and add-global-mode or delete-mode and delete-global-mode commands

137\$ \$cmode and \$gmode variables

138# _cmode

139+ EVariables:cmode

140K \$cmode;\$gmode;mode

\$₁₄₁ #₁₄₂ +₁₄₃ K₁₄₄ \$curchar

This variable contains the ASCII value of the character currently at the point.

141\$ \$curchar variable
142# _curchar
143+ EVariables:curchar
144K \$curchar

`$145 #146 +147 K148 $curcol`

This variable contains the column (starting at 0) of the point in the current buffer.

145\$ \$curcol variable
146# _curcol
147+ EVariables:curcol
148K \$curcol

\$₁₄₉ #₁₅₀ +₁₅₁ K₁₅₂ **\$curline**

This variable contains the line number (starting at 1) of the point in the current buffer.

149\$ \$curline variable
150# _curline
151+ EVariables:curline
152K \$curline

\$₁₅₃ #₁₅₄ +₁₅₅ K₁₅₆ **\$curwidth**

This variable contains the number of columns displayed in the current screen.

Setting this variable is equivalent to using the change-screen-width command with a numeric argument.

153\$ \$curwidth variable
154# _curwidth
155+ EVariables:curwidth
156K \$curwidth

\$₁₅₇ #₁₅₈ +₁₅₉ K₁₆₀ **\$curwind**

This variable contains the index of the current window within the screen. Windows are numbered from top to bottom, starting at 1. The number of windows within the current screen is held by the \$numwind variable.

157\$ \$curwind variable
158# _curwind
159+ EVariables:curwind
160K \$curwind

\$₁₆₁ #₁₆₂ +₁₆₃ K₁₆₄ **\$cwwline**

This variable contains the number of lines displayed in the current window.

161\$ \$cwwline variable
162# _cwwline
163+ EVariables:cwwline
164K \$cwwline

\$₁₆₅ #₁₆₆ +₁₆₇ K₁₆₈ **\$debug**

This boolean variable contains a flag used to trigger macro debugging. If it is set to TRUE, macros are executed step by step, and each statement and variable assignment is displayed on the message line.

Default value: FALSE

165\$ \$debug variable
166# _debug
167+ EVariables:debug
168K \$debug

¹⁶⁹\$ ¹⁷⁰# ¹⁷¹+ ¹⁷²K **\$desktopcolor**

This variable contains the color to use for the desktop. In the MS-Windows version of MicroEMACS, the value of this variable is irrelevant.

Default value: BLACK.

¹⁶⁹\$ \$desktopcolor variable
¹⁷⁰# _desktopcolor
¹⁷¹+ EVariables:desktopcolor
¹⁷²K \$desktopcolor

\$₁₇₃ #₁₇₄ +₁₇₅ K₁₇₆ **\$diagflag**

If this boolean variable is set to TRUE, diagonal dragging of text and mode lines is enabled. If it is FALSE, text and modelines can either be dragged horizontally or vertically but not both at the same time.

173\$ \$diagflag variable
174# _diagflag
175+ EVariables:diagflag
176K \$diagflag

\$₁₇₇ #₁₇₈ +₁₇₉ K₁₈₀ **\$discmd**

If this boolean variable is set to TRUE, the echoing of command prompts and output on the message line is enabled. If it is FALSE, most messages and prompts are disabled (this is handy to avoid some cases of message line flashing while a macro is running).

Default value: TRUE.

177\$ \$discmd variable
178# _discmd
179+ EVariables:discmd
180K \$discmd

\$₁₈₁ #₁₈₂ +₁₈₃ K₁₈₄ \$disinp

If this boolean variable is set to TRUE, the echoing of input at the command prompts is enabled.

Default value: TRUE.

181\$ \$disinp variable
182# _disinp
183+ EVariables:disinp
184K \$disinp

\$₁₈₅ #₁₈₆ +₁₈₇ K₁₈₈ **\$disphigh**

If this boolean variable is set to TRUE, high-bit characters (single byte characters that are greater than 127 in value) will be displayed in a pseudo-control format. The characters "^!" will lead off the sequence, followed by the character stripped of its high bit.

Default value: FALSE.

185\$ \$disphigh variable
186# _disphigh
187+ EVariables:disphigh
188K \$disphigh

\$₁₈₉ #₁₉₀ +₁₉₁ K₁₉₂ \$exbhook

This variable holds the name of a command or macro which is run whenever you are switching out of a buffer.

Default value: nop

189\$ \$exbhook variable
190# _exbhook
191+ EVariables:exbhook
192K \$exbhook;hook;buffer

`$193 #194 +195 K196 $fcol`

This variable contains the line position being displayed in the first column of the current window.

193\$ \$fcol variable
194# _fcol
195+ EVariables:fcol
196K \$fcol

\$₁₉₇ #₁₉₈ +₁₉₉ K₂₀₀ **\$fillcol**

This variable contains the current fill column. It is used by the fill-paragraph command. It can be set through the set command or by using the set-fill-column command.

Default value: 72

197\$ \$fillcol variable
198# _fillcol
199+ EVariables:fillcol
200K \$fillcol;fill

²⁰¹_{\$} ²⁰²_# ²⁰³₊ ²⁰⁴_K **\$flicker**

In the MS-DOS version of MicroEMACS, this variable contains a flicker flag that should be set to TRUE if the display is an IBM CGA and set to FALSE for most others.

In the MS-Windows version of MicroEMACS, this variable can be set to FALSE to allow an animated grinder to be displayed in place of the hourglass mouse cursor. Since this animation results, on many video displays, in an annoying flicker of the cursor, it is disabled when \$flicker is set to TRUE.

Default value: TRUE

²⁰¹_{\$} \$flicker variable

²⁰²_# _flicker

²⁰³₊ EVariables:flicker

²⁰⁴_K \$flicker;grinder;hourglass

²⁰⁵\$ ²⁰⁶# ²⁰⁷+ ²⁰⁸K **\$fmtlead**

A line starting with one of the characters in the \$fmtlead variable is considered to be a text formatter command. Therefore, the following line is considered to be the start of a paragraph.

If you are editing text destined for use by a text formatter, set \$fmtlead to the command character for that formatter. That will prevent MicroEMACS from formatting what should be lines of commands meant for the formatter. If, for example, you are editing SCRIBE source, use the set (^XA) command to set \$fmtlead to "@".

Default value: empty string

205\$ \$fmtlead variable
206# _fmtlead
207+ EVariables:fmtlead
208K \$fmtlead;paragraph

\$₂₀₉ #₂₁₀ +₂₁₁ K₂₁₂ **\$gflags**

Some of the ways MicroEMACS controls its internal functions can be modified by the value in the \$gflags variable. Each bit in this variable will be used to control a different function:

- 1 If this bit is set to zero, EMACS will not automatically switch to the buffer of the first file after executing the startup macros.
- 2 If this bit is set to one, suppress redraw events.

209\$ \$gflags variable
210# _gflags
211+ EVariables:gflags
212K \$gflags

\$₂₁₃ #₂₁₄ +₂₁₅ K₂₁₆ **\$hardtab**

This variable contains the number of spaces between hard tab stops. This can be used to change the way tabs are displayed within the editor.

Default value: 8

213\$ \$hardtab variable
214# _hardtab
215+ EVariables:hardtab
216K \$hardtab

`$217 #218 +219 K220 $highlight`

When this variable contains a value n between 0 and 14, it indicates that the text located between the marks n and $n+1$ should be highlighted. A value of 255 indicates that no highlighting is performed.

Default value: 10

217\$ \$highlight variable
218# _highlight
219+ EVariables:highlight
220K \$highlight

\$₂₂₁ #₂₂₂ +₂₂₃ K₂₂₄ \$hjump

This variable contains the number of columns the editor should scroll the screen horizontally when a horizontal scroll is required.

Default value: 1

221\$ \$hjump variable
222# _hjump
223+ EVariables:hjump
224K \$hjump

\$²²⁵ #²²⁶ +²²⁷ K²²⁸ **\$hscroll**

This variable is a flag that determines if MicroEMACS will scroll the entire window horizontally, or just the current line. The default value, TRUE, results in the entire window being shifted left or right when the cursor goes off the edge of the screen.

225\$ \$hscroll variable
226# _hscroll
227+ EVariables:hscroll
228K \$hscroll

\$₂₂₉ #₂₃₀ +₂₃₁ K₂₃₂ **\$hscrollbar**

This boolean variable exists only under the MS-Windows version of MicroEMACS. If it is TRUE, an horizontal scroll bar is available at the bottom of each screen, allowing you to scroll the text in the current window right and left.

If \$hscrollbar is FALSE, the horizontal scroll bar is not present.

Default value: TRUE

229\$ \$hscrollbar variable
230# _hscrollbar
231+ EVariables:hscrollbar
232K \$hscrollbar;scroll bar

\$₂₃₃ #₂₃₄ +₂₃₅ K₂₃₆ \$istern

This variable contains the character used to terminate incremental search string inputs.

Default value: the last key bound to meta-prefix (initially: Escape character)

233\$ \$istern variable

234# _istern

235+ EVariables:istern

236K \$istern;incremental search

\$₂₃₇ #₂₃₈ +₂₃₉ K₂₄₀ \$kill

This variable contains the first 127 characters currently in the kill buffer.

Attempts to set this variable are ignored.

237\$ \$kill variable

238# _kill

239+ EVariables:kill

240K \$kill

²⁴¹\$ ²⁴²# ²⁴³+ ²⁴⁴K **\$language**

This variable contains the name of the national language in which MicroEMACS messages will be displayed. (Currently MicroEMACS is available in English, French, Spanish, Latin, Portuguese, Dutch, German, and Pig Latin).

The MS-Windows version of MicroEMACS is currently available in English only.

Attempts to set this variable are ignored. Changing the language used by MicroEMACS requires recompiling.

²⁴¹\$ \$language variable

²⁴²# _language

²⁴³+ EVariables:language

²⁴⁴K \$language

\$₂₄₅ #₂₄₆ +₂₄₇ K₂₄₈ **\$lastkey**

This variable contains a number representing the ASCII value of the last key press processed by MicroEMACS. This variable does not contain any indication that the last keystroke was prefixed by the Meta or the **Alt** keys. Further more, function or special keys are perceived as the last character of their keystroke representation.

Note that this variable does not change during playback of a keyboard macro.

Setting this variable does not have any effect on the editor beyond changing the variable's value.

245\$ \$lastkey variable
246# _lastkey
247+ EVariables:lastkey
248K \$lastkey

\$₂₄₉ #₂₅₀ +₂₅₁ K₂₅₂ **\$lastmesg**

This variable contains the text of the last message which MicroEMACS wrote on the message line.

Setting this variable does not have any effect on the editor beyond changing the variable's value.

249\$ \$lastmesg variable
250# _lastmesg
251+ EVariables:lastmesg
252K \$lastmesg

`$253 #254 +255 K256 $line`

This variable contains the first 127 characters of the current line. Setting this variable overwrites the contents of the current line.

253\$ \$line variable
254# _line
255+ EVariables:line
256K \$line

²⁵⁷\$ ²⁵⁸# ²⁵⁹+ ²⁶⁰K \$Iterm

This variable contains the string of characters to use as a line terminator when writing a file to disk. By default, it is an empty string, which causes a newline to be written (under MS-DOS or MS-Windows, this translates into a carriage return character followed by a line feed character).

Under some operating systems, the value of this variable is irrelevant.

²⁵⁷\$ \$Iterm variable
²⁵⁸# _Iterm
²⁵⁹+ EVariables:Iterm
²⁶⁰K \$Iterm

$\$_{261} \#_{262} +_{263} K_{264} \$lwidth$

This variable contains the number of characters of the current line.

Attempts to set this variable are ignored.

261\$ \$lwidth variable
262# _lwidth
263+ EVariables:lwidth
264K \$lwidth

`$265 #266 +267 K268 $match`

This variable contains the last string matched by a search operation.

Attempts to set this variable are ignored.

265\$ \$match variable
266# _match
267+ EVariables:match
268K \$match

\$₂₆₉ #₂₇₀ +₂₇₁ K₂₇₂ \$mmove

If this variable it is equal to 2, any mouse movement results in a mouse action (MSm, S-MSm or MS^m).

If this variable is set to 1, some mouse movement that are of marginal interest (like while a popup buffer is being displayed or, under MS-Windows, while no mouse button is pressed) are ignored.

If \$mmove is set to 0, all mouse movements are ignored.

Default value: 1

269\$ \$mmove variable
270# _mmove
271+ EVariables:mmove
272K \$mmove;mouse

\$₂₇₃ #₂₇₄ +₂₇₅ K₂₇₆ **\$modeflag**

If this boolean variable is TRUE, mode lines are visible. If it is FALSE, mode lines are not displayed (thus allowing one more line per window).

Default value: TRUE

273\$ \$modeflag variable
274# _modeflag
275+ EVariables:modeflag
276K \$modeflag

\$₂₇₇ #₂₇₈ +₂₇₉ K₂₈₀ **\$msflag**

Under some operating systems, this boolean variable can be used to control the use of the pointing device: when it is TRUE, the mouse (if present) is active. When it is FALSE, the mouse cursor is not displayed, and mouse actions are ignored.

Under MS-Windows, setting this variable to FALSE does not cause the cursor to be hidden, but mouse actions within text areas are ignored. However, the mouse remains useable to activate menus or select, move and resize screens.

Default value: TRUE

277\$ \$msflag variable
278# _msflag
279+ EVariables:msflag
280K \$msflag;mouse

\$₂₈₁ #₂₈₂ +₂₈₃ K₂₈₄ \$numwind

This variable contains the number of windows displayed within the current screen.

Attempts to set this variable are ignored.

281\$ \$numwind variable
282# _numwind
283+ EVariables:numwind
284K \$numwind

\$₂₈₅ #₂₈₆ +₂₈₇ K₂₈₈ **\$oldcrypt**

If this boolean variable is TRUE, the CRYPT mode uses the old method of encryption (which had a bug in it). This allows you to read files that were encrypted with a previous version of MicroEMACS.

Default value: FALSE.

285\$ \$oldcrypt variable
286# _oldcrypt
287+ EVariables:oldcrypt
288K \$oldcrypt

\$₂₈₉ #₂₉₀ +₂₉₁ K₂₉₂ **\$orgrow**

This variable contains the position of the current screen's top row on the desktop, starting at 0.

Setting this variable is equivalent to invoking the change-screen-row command.

Under MS-Windows, the value of this variable is irrelevant.

Default value: 0

289\$ \$orgrow variable
290# _orgrow
291+ EVariables:orgrow
292K \$orgrow

`$293 #294 +295 K296 $os`

This variable contains a string that identifies the operating system. It is set to MSWIN in the Microsoft Windows version of MicroEMACS.

Attempts to set this variable are ignored.

293\$ \$os variable
294# _os
295+ EVariables:os
296K \$os

²⁹⁷\$ ²⁹⁸# ²⁹⁹+ ³⁰⁰K **\$orgcol**

This variable contains the position of the current screen's left column on the desktop, starting at 0.

Setting this variable is equivalent to invoking the change-screen-column command.

Under MS-Windows, the value of this variable is irrelevant.

Default value: 0

²⁹⁷\$ \$orgcol variable
²⁹⁸# _orgcol
²⁹⁹+ EVariables:orgcol
³⁰⁰K \$orgcol

`$301 #302 +303 K304 $overlap`

This variable contains the amount of overlapping, in number of lines, used when paging up and down (using the next-page and previous-page commands).

Default value: 2

301\$ \$overlap variable
302# _overlap
303+ EVariables:overlap
304K \$overlap

\$₃₀₅ #₃₀₆ +₃₀₇ K₃₀₈ **\$pagelen**

This variable contains the number of lines (including mode lines) displayed by the current screen.

Setting this variable is equivalent to invoking the change-screen-size command with a numeric argument.

305\$ \$pagelen variable
306# _pagelen
307+ EVariables:pagelen
308K \$pagelen

\$₃₀₉ #₃₁₀ +₃₁₁ K₃₁₂ **\$palette**

This variable contains a string that is used to control the color palette settings on graphics versions of MicroEMACS.

Under MS-Windows, \$palette is composed of up to 48 octal digits. Each group of three digits redefines an entry of the palette, by specifying the red, green and blue levels of that color.

Default value: empty string

309\$ \$palette variable
310# _palette
311+ EVariables:palette
312K \$palette

³¹³\$ ³¹⁴# ³¹⁵+ ³¹⁶K **\$paralead**

A line starting with one of the characters in the \$paralead variable is considered to be the first line of a paragraph.

Default value: Space and TAB characters

³¹³\$ \$paralead variable

³¹⁴# _paralead

³¹⁵+ EVariables:paralead

³¹⁶K \$paralead;paragraph

`$317 #318 +319 K320 $pending`

This boolean variable is TRUE if there are type ahead keystrokes waiting to be processed.

Attempts to set this variable are ignored.

317\$ \$pending variable
318# _pending
319+ EVariables:pending
320K \$pending

\$₃₂₁ #₃₂₂ +₃₂₃ K₃₂₄ **\$popflag**

If this boolean variable is TRUE, popup buffers are used instead of opening a window for building completion lists and by the following commands:

apropos
describe-bindings
describe-functions
describe-variables
list-buffers
list-screens
show-files

Default value: TRUE

321\$ \$popflag variable
322# _popflag
323+ EVariables:popflag
324K \$popflag

\$₃₂₅ #₃₂₆ +₃₂₇ K₃₂₈ **\$posflag**

If this boolean variable is TRUE, the position of the point (row and column) is displayed in the current window's mode line.

Default value: FALSE

325\$ \$posflag variable
326# _posflag
327+ EVariables:posflag
328K \$posflag

\$₃₂₉ #₃₃₀ +₃₃₁ K₃₃₂ **\$progname**

This variable contains the string "MicroEMACS" for standard MicroEMACS. It can be something else if MicroEMACS is incorporated as part of someone else's program.

Attempts to set this variable are ignored. Changing it requires recompiling.

329\$ \$progname variable
330# _progname
331+ EVariables:progname
332K \$progname

\$₃₃₃ #₃₃₄ +₃₃₅ K₃₃₆ **\$readhook**

The command or macro named in this variable is run when a file is read into a buffer. This can be used to implement modes which are specific to a particular file or file type.

Default value: nop

333\$ \$readhook variable
334# _readhook
335+ EVariables:readhook
336K \$readhook;hook;read

\$₃₃₇ #₃₃₈ +₃₃₉ K₃₄₀ **\$region**

This variable contains the first 255 characters of the current region. If the region is not defined (because the mark is not set), this variable contains the string: "ERROR".

Attempts to set this variable are ignored.

337\$ \$region variable
338# _region
339+ EVariables:region
340K \$region

\$₃₄₁ #₃₄₂ +₃₄₃ K₃₄₄ **\$replace**

This variable contains the current default replace string. That is the replace string that was specified in the last replace-string or query-replace-string command and will be used as default value for the next such command.

341\$ \$replace variable
342# _replace
343+ EVariables:replace
344K \$replace;replace

\$³⁴⁵ #³⁴⁶ +³⁴⁷ K³⁴⁸ **\$rval**

This variable contains the returned value from the last subprocess which was invoked from MicroEMACS's commands: execute-program, filter-buffer, i-shell, pipe-command.and shell-command.

Under MS-Windows, this variable always has the value 0.

Attempts to set this variable are ignored.

345\$ \$rval variable
346# _rval
347+ EVariables:rval
348K \$rval

\$₃₄₉ #₃₅₀ +₃₅₁ K₃₅₂ **\$scrname**

This variable contains the current screen's name.

Setting this variable causes the specified screen to be made the current one. If that screen does not exist, nothing happens. To change the name of a screen, use the rename-screen command.

349\$ \$scrname variable
350# _scrname
351+ EVariables:scrname
352K \$scrname

\$₃₅₃ #₃₅₄ +₃₅₅ K₃₅₆ **\$search**

This variable contains the current default search string. That is the search string that was specified in the last search-forward, search-reverse, incremental-search, reverse-incremental-search, replace-string or query-replace-string command and will be used as default value for the next such command or as the target for hunt-forward and hunt-backward.

353\$ \$search variable
354# _search
355+ EVariables:search
356K \$search;search;replace

\$³⁵⁷ #³⁵⁸ +³⁵⁹ K³⁶⁰ **\$searchpnt**

The value of this variable specifies the positioning of the of the point at the end of a successful search:

- If \$searchpnt = 0, the cursor is placed at the end of the matched text on forward searches, and at the beginning of this text on reverse searches.
- If \$searchpnt = 1, the cursor is placed at the beginning of the matched text regardless of the search direction.
- If \$searchpnt = 2, the cursor is placed at the end of the matched text regardless of the search direction.

Setting this variable to a value other than one of the above causes the value 0 to be used.

Default value: 0

357\$ \$searchpnt variable
358# _searchpnt
359+ EVariables:searchpnt
360K \$searchpnt

`$361 #362 +363 K364 $seed`

This variable contains the integer seed of the random number generator. This is used by the &rnd function and also to compute temporary file names (if \$ssave is TRUE).

Initial value: 0

361\$ \$seed variable
362# _seed
363+ EVariables:seed
364K \$seed

\$₃₆₅ #₃₆₆ +₃₆₇ K₃₆₈ **\$softtab**

The value of this variable relates to the number of spaces inserted by MicroEMACS when the handle-tab command (which is normally bound to the TAB key) is invoked:

If \$softtab is n , strictly positive, tabs stops are located at every n^{th} column and the handle-tab command inserts space characters in sufficient number to move the point to the next tab stop.

If \$softtab is zero, the handle-tab command inserts true tab characters.

If \$softtab is strictly negative, the handle-tab command fails.

This variable can be set by passing a numeric argument to handle-tab or by directly using the set command.

Default value: 0

365\$ \$softtab variable
366# _softtab
367+ EVariables:softtab
368K \$softtab

\$₃₆₉ #₃₇₀ +₃₇₁ K₃₇₂ **\$sres**

This variable contains a string that identifies the current screen resolution (CGA, MONO, EGA or VGA on the IBM-PC, LOW, MEDIUM, HIGH or DENSE on the Atari ST1040, MSWIN under Microsoft Windows and NORMAL on most others).

Depending on the hardware and operating system MicroEMACS is running on, setting this variable may allow you to change the screen resolution. Not that under MS-Windows, attempts to set this variable are ignored.

369\$ \$sres variable
370# _sres
371+ EVariables:sres
372K \$sres

\$373 #374 +375 K376 **\$ssave**

If this boolean variable is TRUE, MicroEMACS perform "safe saves": when it is asked to save the current buffer to disk, it writes it out to a temporary file, deletes the original file, and then renames the temporary to the old file name.

If \$ssave is FALSE, MicroEMACS performs saves by directly overwriting the original file, thus risking loss of data if a system crash occurs before the end of the save operation. On the other hand, this mode insures that the original file attributes (ownership and access rights) are preserved on systems that support these (like UNIX).

Default value: TRUE.

373\$ \$ssave variable
374# _ssave
375+ EVariables:ssave
376K \$ssave

\$₃₇₇ #₃₇₈ +₃₇₉ K₃₈₀ **\$sscroll**

If this boolean variable is TRUE, MicroEMACS is configured for smooth vertical scrolling: when the cursor moves off the top or bottom of the current window, the window's contents scroll up or down one line at a time.

If \$sscroll is FALSE, scrolling occurs by half pages.

Default value: FALSE

377\$ \$sscroll variable
378# _sscroll
379+ EVariables:sscroll
380K \$sscroll

\$₃₈₁ #₃₈₂ +₃₈₃ K₃₈₄ **\$status**

This boolean variable contains the status returned by the last command. This is usually used with the !FORCE directive to check on the success of a search, or a file operation.

Setting this variable can be used to return a FALSE status from a macro.

381\$ \$status variable
382# _status
383+ EVariables:status
384K \$status

\$₃₈₅ #₃₈₆ +₃₈₇ K₃₈₈ **\$stern**

This variable contains the character used to terminate search string inputs.

Default value: the last key bound to meta-prefix (initially: Escape character)

385\$ \$stern variable
386# _stern
387+ EVariables:stern
388K \$stern;replace;search

\$₃₈₉ #₃₉₀ +₃₉₁ K₃₉₂ **\$target**

This variable contains the column position where the point will attempt to move after a next-line or previous-line command. Unless the previous command was next-line or previous-line, the default value for this variable is the current column.

389\$ \$target variable
390# _target
391+ EVariables:target
392K \$target

³⁹³\$ ³⁹⁴# ³⁹⁵+ ³⁹⁶K **\$time**

This variable contains a string corresponding to the current date and time. Usually this is given in a form like to "Mon May 09 10:10:58 1988". Not all operating systems support this.

³⁹³\$ \$time variable
³⁹⁴# _time
³⁹⁵+ EVariables:time
³⁹⁶K \$time

\$₃₉₇ #₃₉₈ +₃₉₉ K₄₀₀ **\$timeflag**

If this boolean variable is TRUE, the current time is displayed on the bottom mode line of each screen.

Default value: FALSE.

Note: Under MS-Windows, this feature currently does not operate properly because MicroEMACS makes incorrect assumptions about the format of the time string (see \$time).

397\$ \$timeflag variable
398# _timeflag
399+ EVariables:timeflag
400K \$timeflag

\$₄₀₁ #₄₀₂ +₄₀₃ K₄₀₄ **\$tpause**

This variable contains the length of the pause used to show a matched fence when the current buffer is in CMODE and a closing fence (a character among "}}") has been typed.

On most systems, this pause is performed by a CPU loop and therefore, the value of \$tpause may need to be adjusted to compensate for the processor's speed.

Under MS-Windows, the pause is performed by a bona-fide timer and \$tpause is expressed in milliseconds. The default value is 1000.

401\$ \$tpause variable
402# _tpause
403+ EVariables:tpause
404K \$tpause

`$405 #406 +407 K408 $version`

This variable contains the current MicroEMACS version number (i.e. "3.11c").

Attempts to set this variable are ignored.

405\$ \$version variable
406# _version
407+ EVariables:version
408K \$version

\$₄₀₉ #₄₁₀ +₄₁₁ K₄₁₂ **\$vscrollbar**

This boolean variable exists only under the MS-Windows version of MicroEMACS. If it is TRUE, a vertical scroll bar is available at the right end of each screen, allowing you to scroll the text in the current window up and down.

If \$vscrollbar is FALSE, the vertical scroll bar is not present.

Default value: TRUE

409\$ \$vscrollbar variable
410# _vscrollbar
411+ EVariables:vscrollbar
412K \$vscrollbar;scroll bar

\$₄₁₃ #₄₁₄ +₄₁₅ K₄₁₆ **\$wchars**

This variable is used to define what a word is for MicroEMACS. It contains the list of all the characters that can be considered part of a word.

If \$wchar is empty, a word is defined as composed of upper and lower case letters, numerals (0 to 9) and the underscore character.

Default value: empty

413\$ \$wchars variable
414# _wchars
415+ EVariables:wchars
416K \$wchars;word

\$₄₁₇ #₄₁₈ +₄₁₉ K₄₂₀ **\$wline**

This variable contains the number of lines displayed in the current window, excluding the mode line.

Setting this variable is equivalent to using the resize-window command with a numeric argument.

417\$ \$wline variable
418# _wline
419+ EVariables:wline
420K \$wline

\$₄₂₁ #₄₂₂ +₄₂₃ K₄₂₄ **\$wraphook**

This variable contains the name of a command or macro which is executed when a buffer is in WRAP mode and it is time to wrap the current line.

Default value: wrap-word

421\$ \$wraphook variable
422# _wraphook
423+ EVariables:wraphook
424K \$wraphook;hook;wrap

\$₄₂₅ #₄₂₆ +₄₂₇ K₄₂₈ **\$writehook**

This variable contains the name of a command or macro which is invoked whenever MicroEMACS attempts to write a file out to disk. This is executed before the file is written, allowing you to process a file on the way out.

Default value: nop

425\$ \$writehook variable
426# _writehook
427+ EVariables:writehook
428K \$writehook;hook;write

\$₄₂₉ #₄₃₀ +₄₃₁ K₄₃₂ **\$xpos**

This variable contains the horizontal screen coordinate where the mouse was located the last time a mouse button was pressed or released.

The leftmost column is considered to be 0 in screen coordinates.

429\$ \$xpos variable
430# _xpos
431+ EVariables:xpos
432K \$xpos;mouse

\$₄₃₃ #₄₃₄ +₄₃₅ K₄₃₆ **\$yankflag**

This boolean variable controls the placement of the point after a yank, yank-pop, insert-file or insert-clip command.

If \$yankflag is FALSE, the point is moved to the end of the yanked or inserted text.

If \$yankflag is TRUE, the cursor remains at the start of the yanked or inserted text.

Default value: FALSE

433^{\$} \$yankflag variable
434[#] _yankflag
435⁺ EVariables:yankflag
436^K \$yankflag

\$₄₃₇ #₄₃₈ +₄₃₉ K₄₄₀ **\$ypos**

This variable contains the vertical screen coordinate where the mouse was located the last time a mouse button was pressed or released.

The top row is considered to be 0 in screen coordinates.

437\$ \$ypos variable
438# _ypos
439+ EVariables:ypos
440K \$ypos;mouse

\$⁴⁴¹ #⁴⁴² +⁴⁴³ K⁴⁴⁴ Interactive Variables

Interactive variables are actually a method to prompt the user for a string. This is done by using an at sign "@" followed with a string argument. The string is displayed on the message line, and the editor waits for the user to type in a string which is then returned as the value of the interactive variable. For example:

```
find-file @"What file? "
```

will ask the user for a file name, and then attempt to find it. Note also that complex expressions can be built up with these operators, such as:

```
set %default "file1"  
@&cat &cat "File to decode[" %default "]: "
```

which prompts the user with the string:

```
File to decode[file1]:
```

441\$ Interactive Variables

442# InteractiveVariables

443+ Variables:interactivevariables

444K interactive;variable

\$⁴⁴⁵ #⁴⁴⁶ +⁴⁴⁷ K⁴⁴⁸ User Variables

User variables allow you to store strings and manipulate them. These strings can be pieces of text, numbers (in text form), or the logical values TRUE and FALSE. These variables can be combined, tested, inserted into buffers, and otherwise used to control the way your macros execute. Up to 512 user variables may be in use in one editing session. All user variable names must begin with a percent sign "%" and may contain any printing character. Only the first 10 characters are significant (i.e. differences beyond the tenth character are ignored).

When a user variable has not been set, it has the value: "ERROR".

445\$ User Variables

446# UserVariables

447+ Variables:uservariables

448K variable

Functions are part of the MicroEMACS Macro language. They can be used wherever an argument (number, string or boolean) is needed.

Function names always begin with the ampersand "&" character, and only the first three characters after the ampersand are significant. Functions are always used in lower case.

Functions can be used to act on variables in various ways. Functions can have one, two, or three arguments. These are always placed after the function, and they can include functions (with their own arguments).

By topic:

Boolean functions
Numeric functions
String functions
Miscellaneous functions

By returned value:

Boolean:	<u>&and</u> , <u>&equal</u> , <u>&exist</u> , <u>&greater</u> , <u>&isnum</u> , <u>&less</u> , <u>&not</u> , <u>&or</u> , <u>&sequal</u> , <u>&sgreater</u> and <u>&sless</u>
Numeric:	<u>&abs</u> , <u>&add</u> , <u>&ascii</u> , <u>&band</u> , <u>&bnot</u> , <u>&bor</u> , <u>&bxor</u> , <u>&divide</u> , <u>&length</u> , <u>&mod</u> , <u>&negate</u> , <u>&rnd</u> , <u>&sindex</u> , <u>&sub</u> and <u>&times</u>
String:	<u>&bind</u> , <u>&cat</u> , <u>&chr</u> , <u>&env</u> , <u>&find</u> , <u>&group</u> , <u>&gtc</u> , <u>&gtk</u> , <u>&indirect</u> , <u>&left</u> , <u>&lower</u> , <u>&mid</u> , <u>&rev</u> , <u>&right</u> , <u>&slower</u> , <u>&supper</u> , <u>&trim</u> , <u>&upper</u> and <u>&xlate</u>

\$₄₅₃ #₄₅₄ +₄₅₅ K₄₅₆ **Boolean Functions**

These functions perform operations on boolean arguments:

&and	<i>log1</i>	<i>log2</i>	Returns TRUE if both boolean arguments are TRUE
&not	<i>log</i>		Returns the opposite boolean value
&or	<i>log1</i>	<i>log2</i>	Returns TRUE if either argument is TRUE

453\$ Boolean Functions

454# BooleanFunctions

455+ Functions:booleanfunctions

456K function;∧¬&or

\$₄₅₇ #₄₅₈ +₄₅₉ K₄₆₀ Numeric Functions

These functions perform operations on numerical arguments:

&abs	<i>num</i>	Returns the absolute value of <i>num</i>
&add	<i>num1 num2</i>	Adds two numbers
&band	<i>num1 num2</i>	Bitwise AND function
&bnot	<i>num</i>	Bitwise NOT function
&bor	<i>num1 num2</i>	Bitwise OR function
&bxor	<i>num1 num2</i>	Bitwise XOR function
&chr	<i>num</i>	Returns a string with the character represented by ASCII code <i>num</i> . This function is the opposite of <u>&ascii</u>
&divide	<i>num1 num2</i>	Divides <i>num1</i> by <i>num2</i> , giving an integer result
&equal	<i>num1 num2</i>	Returns TRUE if <i>num1</i> and <i>num2</i> are numerically equal
&greater	<i>num1 num2</i>	Returns TRUE if <i>num1</i> is greater than, or equal to <i>num2</i>
&isnum	<i>num</i>	Returns TRUE if the given argument is a legitimate number
&less	<i>num1 num2</i>	Returns TRUE if <i>num1</i> is less than <i>num2</i>
&mod	<i>num1 num2</i>	Returns the remainder of dividing <i>num1</i> by <i>num2</i>
&negate	<i>num</i>	Multiplies <i>num</i> by -1
&rnd	<i>num</i>	Returns a random integer between 1 and <i>num</i>
&sub	<i>num1 num2</i>	Subtracts <i>num2</i> from <i>num1</i>
&times	<i>num1 num2</i>	Multiplies <i>num1</i> by <i>num2</i>

457\$ Numeric Functions

458# NumericFunctions

459+ Functions:numericfunctions

460K

function;&abs;&add;&band;⌐&bor;&bxor;&chr;÷&equal;&greater;&isnum;&less;&mod;
&negate;⊂×

\$₄₆₁ #₄₆₂ +₄₆₃ K₄₆₄ String Functions

These functions perform operations related to strings. All of them have at least one string argument:

&ascii	<i>str</i>	Returns the ASCII code of the first character in <i>str</i> . This function is the opposite of <u>&chr</u>
&cat	<i>str1 str2</i>	Concatenates the two strings to form one
&indirect	<i>str</i>	Evaluate <i>str</i> as a variable.
&left	<i>str num</i>	Returns the <i>num</i> leftmost characters from <i>str</i>
&length	<i>str</i>	Returns length of string
&lower	<i>str</i>	Transforms <i>str</i> to lowercase
&mid	<i>str num1 num2</i>	Starting from <i>num1</i> position in <i>str</i> , returns <i>num2</i> characters
&rev	<i>str</i>	Reverses the order of characters in <i>str</i>
&right	<i>str num</i>	Returns the <i>num</i> rightmost characters from <i>str</i>
&sequal	<i>str1 str2</i>	Returns TRUE if the two strings are the same
&sgreater	<i>str1 str2</i>	Returns TRUE if <i>str1</i> is alphabetically greater than or equal to <i>str2</i>
&sindex	<i>str1 str2</i>	Returns the position of <i>str2</i> within <i>str1</i> . Returns zero if not found
&sless	<i>str1 str2</i>	Returns TRUE if <i>str1</i> is less alphabetically than <i>str2</i>
&slower	<i>str1 str2</i>	Translate the first char in <i>str1</i> to the first char in <i>str2</i> when lowercasing.
&supper	<i>str1 str2</i>	Translate the first char in <i>str1</i> to the first char in <i>str2</i> when uppercasing.
&trim	<i>str</i>	Trims the trailing white space from a string
&upper	<i>str</i>	Transforms <i>str</i> to uppercase
&xlate	<i>source lookup trans</i>	Translate each character of <i>source</i> that appears in <i>lookup</i> to the corresponding character from <i>trans</i>

461\$ String Functions

462# StringFunctions

463+ Functions:stringfunctions

464K

function;&ascii;&cat;&left;&length;&lower;∣&right;&sequal;&sgreater;&sindex;&sless;&slower;&supper;&trim;&upper;&xlate

\$₄₆₅ #₄₆₆ +₄₆₇ K₄₆₈

Miscellaneous Functions

&bind	<i>str</i>	Returns the name of the command bound to the keystroke <i>str</i>
&env	<i>str</i>	If the operating system has this capability, this returns the environment string associated with <i>str</i>
&exist	<i>str</i>	Returns TRUE if the named file <i>str</i> exists
&find	<i>str</i>	Finds the named file <i>str</i> along the <u>path</u> and return its full file specification or an empty string if no such file exists
&group	<i>num</i>	Return <u>group</u> <i>num</i> as set by a <u>MAGIC</u> mode search.
&gt;c		Returns a string of characters containing a MicroEMACS command input from the user
&g;k		Returns a string containing a single keystroke from the user

465\$ Miscellaneous Functions

466# MiscellaneousFunctions

467+ Functions:zzzmiscellaneousfunctions

468K function;&bind;&env;∃&find;&group;>c;&g;k

\$₄₆₉ #₄₇₀ +₄₇₁ K₄₇₂ **&indirect**

The **&indirect** function evaluates its argument, takes the resulting string, and then uses it as a variable name. For example, given the following piece of macro language:

```
; set up reference table
set %one "elephant"
set %two "giraffe"
set %three "donkey"
set %index "%two"
insert-string &ind %index
```

The string "giraffe" would have been inserted at the point in the current buffer. This indirection can be safely nested up to about 10 levels.

469\$ &indirect function
470# .indirect
471+ StringFunctions:indirect
472K &indirect;function

\$₄₇₃ #₄₇₄ +₄₇₅ K₄₇₆ **Comments**

Within the macro language, a semicolon ";" signals the beginning of a comment. The text from the semicolon to the end of the line is ignored by MicroEMACS.

A comment can be the only content of a line, in which case the semicolon must be the first non-blank character on the line. A comment can also appear at the end of any statement.

Note that empty lines are legal (treated as comments).

473\$ Comments

474# Comments

475+ MacroLanguage:070

476K comment