$[1] #[2] +[3] K[4] 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} K^8
Commands

By topic:

5[$] Commands
6[#] Commands
7[+] MacroLanguage:010
8[K] commands

$[9] #[10] +[11] K[12] **Binding commands**

- apropos
- bind-to-key
- bind-to-menu
- ctlx-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
12K binding;commands

$[13] #[14] +[15] K[16] 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
- detab-region
- entab-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] K[20] 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:bufferwindowscreencommands
20K buffer;window;screen;commands

$[21] #[22] +[23] K[24] 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
24[K] clipboard;kill;commands

$[25] #[26] +[27] K[28] 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] K[32] **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] K[36] 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] K[40] 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] K[44] 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] K[48] 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] K[52] Standard commands

The following commands are available in all implementations of MicroEMACS:

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

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

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

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

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

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

$[53] #[54] +[55] K[56] Additional commands

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

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

53$ Additional commands
54# AdditionalCommands
55+ CommandsByTopic:zzz910
56K Additionnal commands;Microsoft Windows;MS Windows

$[57] #[58] +[59] K[60] 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
60[K] directives;macro

$[61] #[62] +[63] K[64] !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] K[68] !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] K[72] **!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] K[76] **!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] K[80] **!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
80[K] !IF;!ELSE;!ENDIF

$[81] #[82] +[83] K[84] **!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] K[88] !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
88[K] !WHILE;!ENDWHILE;loop;!BREAK

$[89] #[90] +[91] K[92]
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 be supplied on the line, the command fails.

89[$] Arguments
90[#] Arguments
91[+] MacroLanguage:030
92[K] argument;evaluate;expression;

$[93] #[94] +[95] K[96] Constants

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.

93$ Constants
94# Constants
95+ MacroLanguage:040
96K constants;macro

\$[97] #[98] +[99] K[100] 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
100[K] variable;macro

$[101] #[102] +[103] K[104] 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] K[108] 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:

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

[105] $ Environmental Variables
[106] # EnvironmentalVariables
[107] + Variables:environmentalvariables
[108] K variable

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

$[109] #[110] +[111] K[112] **$acount**

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[$] $acount variable
110[#] _acount
111[+] EVariables:acount
112[K] $acount;ASAVE;autosave

$[113] #[114] +[115] K[116] **$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

$[117] #[118] +[119] K[120] **$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

$[121] #[122] +[123] K[124] **$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
124[K] $cbflags

$[125] #[126] +[127] K[128] **$cbufname**

This variable contains the name of the current buffer.

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

$[129] #[130] +[131] K[132] **$cfname**

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

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

$[133] #[134] +[135] K[136] **$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

$[137] #[138] +[139] K[140] **$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:

WRAP	1	Word wrap
CMODE	2	C indentation and fence matching
SPELL	4	Interactive spell checking (Not implemented yet)
EXACT	8	Exact matching for searches
VIEW	16	Read-only buffer
OVER	32	Overwrite mode
MAGIC	64	Regular expressions in search
CRYPT	128	Encryption mode active
ASAVE	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
140[K] $cmode;$gmode;mode

$[141] #[142] +[143] K[144] **$curchar**

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

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

$[145] #[146] +[147] K[148] **$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

$[149] #[150] +[151] K[152] **$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] K[156] **$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
156[K] $curwidth

$[157] #[158] +[159] K[160] **$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

$[161] #[162] +[163] K[164] **$cwline**

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

161$ $cwline variable
162# _cwline
163+ EVariables:cwline
164K $cwline

$165 #166 +167 K168 **$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

$[169] #[170] +[171] K[172] **$deskcolor**

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$ $deskcolor variable
170# _deskcolor
171+ EVariables:deskcolor
172K $deskcolor

$[173] #[174] +[175] K[176] **$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

$[177] #[178] +[179] K[180] **$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
180[K] $discmd

\$[181] #[182] +[183] K[184] **\$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] K[188] **$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
188[K] $disphigh

$[189] #[190] +[191] K[192] **$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

$[193] #[194] +[195] K[196] **$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

[$]197 [#]198 [+]199 [K]200 **$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
200[K] $fillcol;fill

$[201] #[202] +[203] K[204] **$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

$[205] #[206] +[207] K[208] **$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

$[209] #[210] +[211] K[212] **$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

$[213] #[214] +[215] K[216] **$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

$[217] #[218] +[219] K[220] **$hilight**

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$ $hilight variable
218# _hilight
219+ EVariables:hilight
220K $hilight

$221 #222 +223 K224 **$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

$[225] #[226] +[227] K[228] **$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
228[K] $hscroll

$[229] #[230] +[231] K[232] **$hscrlbar**

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 $hscrlbar is FALSE, the horizontal scroll bar is not present.

Default value: TRUE

229[$] $hscrlbar variable
230[#] _hscrlbar
231[+] EVariables:hscrlbar
232[K] $hscrlbar;scroll bar

$[233] #[234] +[235] K[236] **$isterm**

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[$] $isterm variable
234[#] _isterm
235[+] EVariables:isterm
236[K] $isterm;incremental search

$[237] #[238] +[239] K[240] **$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

$[241] #[242] +[243] K[244] **$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

$[245] #[246] +[247] K[248] **$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

$[249] #[250] +[251] K[252] **$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

$[253] #[254] +[255] K[256] **$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
256[K] $line

$[257] #[258] +[259] K[260] $lterm

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[$] $lterm variable
258[#] _lterm
259[+] EVariables:lterm
260[K] $lterm

$[261] #[262] +[263] K[264] **$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
264[K] $lwidth

$[265] #[266] +[267] K[268] **$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
268[K] $match

$[269] #[270] +[271] K[272] **$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

$[273] #[274] +[275] K[276] **$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

$[277] #[278] +[279] K[280] **$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

$[281] #[282] +[283] K[284] **$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
284[K] $numwind

$[285] #[286] +[287] K[288] **$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

$[289] #[290] +[291] K[292] **$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

$[293] #[294] +[295] K[296] **$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] K[300] **$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

$[301] #[302] +[303] K[304] **$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

$[305] #[306] +[307] K[308] **$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

$[309] #[310] +[311] K[312] **$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
312[K] $palette

$[313] #[314] +[315] K[316] **$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
316[K] $paralead;paragraph

$[317] #[318] +[319] K[320] **$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
320[K] $pending

$[321] #[322] +[323] K[324] **$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

$[325] #[326] +[327] K[328] **$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

$[329] #[330] +[331] K[332] **$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] K[336] **\$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

$[337] #[338] +[339] K[340] **$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

$[341] #[342] +[343] K[344] **$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
344[K] $replace;replace

$[345] #[346] +[347] K[348] **$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] K[352] **\$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

$[353] #[354] +[355] K[356] **$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] K[360] **$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

$[361] #[362] +[363] K[364] **$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

$[365] #[366] +[367] K[368] **$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
368[K] $softtab

$[369] #[370] +[371] K[372] **$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
372[K] $sres

$[373] #[374] +[375] K[376] **$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

$[377] #[378] +[379] K[380] **$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
380[K] $sscroll

\$[381] #[382] +[383] K[384] **$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
384[K] $status

$[385] #[386] +[387] K[388] **$sterm**

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

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

385[$] $sterm variable
386[#] _sterm
387[+] EVariables:sterm
388[K] $sterm;replace;search

\$[389] #[390] +[391] K[392] **\$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

$[393] #[394] +[395] K[396] **$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

$[397] #[398] +[399] K[400] **$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
400[K] $timeflag

$[401] #[402] +[403] K[404] **$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

$[405] #[406] +[407] K[408] **$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
408[K] $version

$[409] #[410] +[411] K[412] **$vscrlbar**

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 $vscrlbar is FALSE, the vertical scroll bar is not present.

Default value: TRUE

409$ $vscrlbar variable
410# _vscrlbar
411+ EVariables:vscrlbar
412K $vscrlbar;scroll bar

$[413] #[414] +[415] K[416] **$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
416[K] $wchars;word

$[417] #[418] +[419] K[420] $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

$[421] #[422] +[423] K[424] **$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
[424]K $wraphook;hook;wrap

$[425] #[426] +[427] K[428] **$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
[428]K $writehook;hook;write

$[429] #[430] +[431] K[432] **$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] K[436] $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
436K $yankflag

\$[437] #[438] +[439] K[440] **$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] K[444] 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] K[448] 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

$[449] #[450] +[451] K[452] Functions

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

449$ Functions
450# Functions
451+ MacroLanguage:060
452K function

$[453] #[454] +[455] K[456] Boolean Functions

These functions perform operations on boolean arguments:

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

453$ Boolean Functions
454# BooleanFunctions
455+ Functions:booleanfunctions
456K function;&and;&not;&or

\$[457] #[458] +[459] K[460] Numeric Functions

These functions perform operations on numerical arguments:

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

457[\$] Numeric Functions
458[#] NumericFunctions
459[+] Functions:numericfunctions
460[K] function;&abs;&add;&band;&bnot;&bor;&bxor;&chr;&divide;&equal;&greater;&isnum;&less;&mod;&negate;&sub;&times

$[461] #[462] +[463] K[464] String Functions

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

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

461$ String Functions
462# StringFunctions
463+ Functions:stringfunctions
464K function;&ascii;&cat;&left;&length;&lower;&mid;&right;&sequal;&sgreater;&sindex;&sless;&slower;&supper;&trim;&upper;&xlate

$[465] #[466] +[467] K[468] Miscellaneous Functions

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

465$ Miscellaneous Functions
466# MiscellaneousFunctions
467+ Functions:zzzmiscellaneousfunctions
468K function;&bind;&env;&exist;&find;&group;&gtc;&gtk

$[469] #[470] +[471] K[472] **&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] K[476] 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