$_1$ #$_2$ +$_3$ K$_4$ **Macro Language**

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

<u>Commands</u>   manipulate text, buffers, windows, etc... within the editor
<u>Directives</u>   control the flow of execution within a macro
<u>Arguments</u>:
    <u>Constants</u>
    <u>Variables</u>
    <u>Functions</u>
<u>Comments</u>

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

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

1$^\$$ Macro Language
2$^\#$ MacroLanguage
3$^+$ Index:2040
4^K macro;language

$_5$ #$_6$ +$_7$ K$_8$ Commands

By topic:

Alphabetical lists:

5$^\$$ Commands
6$^\#$ Commands
7$^+$ MacroLanguage:010
8^K commands

$^{$}$9 #10 +11 K12 **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
16^{K} 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
20[K] 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
24[K] clipboard;kill;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
28^K execute;spawn;filter;pipe;shell;DOS;macro;variable;commands

File Commands

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

$³³ #₃₄ +₃₅ K₃₆ **Mouse commands**

[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
40^K 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
44^K 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](#)

$^{\$}$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 <u>^X</u> 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

---

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

delete-blank-lines	Delete all blank lines around the point
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-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
56^K Additionnal commands;Microsoft Windows;MS Windows

Directives

Directives are used within <u>macros</u> 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:

<u>!BREAK</u>
<u>!ENDM</u>
<u>!FORCE</u>
<u>!GOTO</u>
<u>!IF</u>, <u>!ELSE</u> and <u>!ENDIF</u>
<u>!RETURN</u>
<u>!WHILE</u> and <u>!ENDWHILE</u>

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 <u>execute-command-line</u> command) are ignored.

$_{61}$ #$_{62}$ $+_{63}$ K$_{64}$ **!BREAK**

This <u>directive</u> lets you abort out of the most inner currently executing <u>while loop</u>, in a <u>macro</u>. 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
64^K !BREAK;!WHILE;loop

$_{65} #_{66} +_{67} K_{68} **!ENDM**

This <u>directive</u> 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:

    <u>store-procedure</u>
    <u>store-macro</u>.

65[$] !ENDM directive
66[#] .ENDM
67[+] Directives:endm
68[K] !ENDM;macro

!FORCE

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

This is often used together with the <u>$status</u> 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
72[K] !FORCE;error

!GOTO

The flow of execution within a MicroEMACS <u>macro</u> can be controlled using the !GOTO <u>directive</u>. 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 <u>!WHILE</u> are usually more efficient than those constructed purely by !GOTOs.

!IF, !ELSE and !ENDIF

The !IF <u>directive</u> allows for conditional execution within a <u>macro</u>.

Lines following the !IF directive, until the corresponding !ELSE or !ENDIF, are executed only if the expression within the !IF line <u>evaluates</u> 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
```

$\$_{81}$ #$_{82}$ +$_{83}$ K$_{84}$ **!RETURN**

This <u>directive</u> causes the current <u>macro</u> 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
84^K !RETURN

$_{85}$ #$_{86}$ +$_{87}$ K$_{88}$ **!WHILE and !ENDWHILE**

This pair of <u>directives</u> facilitates repetitive execution within a <u>macro</u>. If a group of statements needs to be executed while a certain expression <u>evaluates</u> 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 <u>!GOTOs</u> with no ill effects. Using a while loop to enclose a repeated task will run much faster than the corresponding construct using <u>!IFs</u>.

85[$] !WHILE and !ENDWHILE directives
86[#] .WHILE
87[+] Directives:while
88[K] !WHILE;!ENDWHILE;loop;!BREAK

Arguments

In the MicroEMACS <u>macro language</u>, <u>commands</u> and <u>functions</u> 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 <u>numeric arguments</u> 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
92K argument;evaluate;expression;

Constants

Wherever <u>macro language</u> statements need to have <u>arguments</u>, 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 <u>commands</u>, <u>directives</u>, <u>variables</u>, or <u>functions</u>. 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 (<u>^H</u>)
~f	formfeed (<u>^L</u>)
~l	linefeed (<u>^J</u>)
~n	newline
~r	carriage return (<u>^M</u>)
~t	tab (<u>^I</u>)

Any character not in the above table which follows a tilde will be passed unmodified. This action is similar to the <u>quote-character</u> (^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 <u>Macro language</u>. They can be used wherever an <u>argument</u> (number, boolean or string) is needed.

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

97$^\$$ Variables
98$^\#$ Variables
99$^+$ MacroLanguage:050
100^K variable;macro

Buffer Variables

Buffer <u>variables</u> 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
> **B**loomington
> 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
> **I**ndianapolis
> Gary

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

$^{$}$₁₀₅ #₁₀₆ +₁₀₇ K₁₀₈ **Environmental Variables**

These <u>variables</u> 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>$curchar</u>	ASCII value of character
<u>$curcol</u>	Current column
<u>$curline</u>	Current line
<u>$curwidth</u>	Number of columns
<u>$curwind</u>	Window index
<u>$cwline</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>$hjump</u>	Horizontal scrolling quantum
<u>$hscroll</u>	Horizontal scrolling flag
<u>$hscrlbar</u>	Horizontal scroll bar flag
<u>$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>$match</u>	Last string matched in a search
<u>$modeflag</u>	Mode line display flag
<u>$msflag</u>	Mouse flag
<u>$numwind</u>	Number of windows
<u>$oldcrypt</u>	Encryption method flag
<u>$orgrow</u>	Row of current screen within desktop

105$^{$}$ Environmental Variables
106$^{#}$ EnvironmentalVariables
107^{+} Variables:environmentalvariables
108^{K} variable

$orgcol	Column of current screen within desktop
$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 K₁₁₂ **$acount**

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

Initial value: 256

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

$\$_{113}$ #$_{114}$ +$_{115}$ K$_{116}$ **\$asave**

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

Default value: 256

113$^\$$ \$asave variable
114$^\#$ _asave
115$^+$ EVariables:asave
116^K \$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

$\$_{121}$ $\#_{122}$ $+_{123}$ K_{124} **$cbflags**

This <u>variable</u> contains the current <u>buffer</u>'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 <u>narrowed</u>

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.

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

This <u>variable</u> contains the name of the current <u>buffer</u>.

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

This <u>variable</u> contains the file name associated to the current <u>buffer</u>.

^{133} #$_{134}$ +$_{135}$ K$_{136}$ **$cmdhook**

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

Default value: <u>nop</u>

133$^{\$}$ $cmdhook variable
134$^{\#}$ _cmdhook
135^{+} EVariables:cmdhook
136^{K} $cmdhook;hook;keyboard

$cmode and **$gmode**

The two <u>variables</u> $cmode and $gmode contain a number that corresponds to the <u>modes</u> for the current <u>buffer</u> ($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
SPELL	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 <u>add-mode</u> and <u>add-global-mode</u> or <u>delete-mode</u> and <u>delete-global-mode</u> commands

137$^\$$ $cmode and $gmode variables
138$^\#$ _cmode
139$^+$ EVariables:cmode
140^K $cmode;$gmode;mode

$_{141}$ #$_{142}$ +$_{143}$ K$_{144}$ **$curchar**

This <u>variable</u> contains the ASCII value of the character currently at the <u>point</u>.

$_{145}$ #$_{146}$ +$_{147}$ K$_{148}$ **$curcol**

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

$_{149}$ #$_{150}$ +$_{151}$ K$_{152}$ **$curline**

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

$₁₅₃ #₁₅₄ +₁₅₅ K₁₅₆ **$curwidth**

This <u>variable</u> contains the number of columns displayed in the current <u>screen</u>.

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

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

$_{157}$ #$_{158}$ +$_{159}$ K$_{160}$ **$curwind**

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

$_{161} #_{162} +_{163} K_{164} **$cwline**

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

$\$_{165}$ #$_{166}$ +$_{167}$ K$_{168}$ **$debug**

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

Default value: FALSE

$₁₆₉ #₁₇₀ +₁₇₁ K₁₇₂ **$deskcolor**

This <u>variable</u> 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
172[K] $deskcolor

$diagflag

If this boolean <u>variable</u> is set to TRUE, diagonal <u>dragging</u> 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.

$_{177}$ #$_{178}$ +$_{179}$ K$_{180}$ **$discmd**

If this boolean <u>variable</u> is set to TRUE, the echoing of command prompts and output on the <u>message line</u> 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 <u>variable</u> 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
184^{K} $disinp

$disphigh

If this boolean <u>variable</u> 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 <u>command</u> or <u>macro</u> which is run whenever you are switching out of a <u>buffer</u>.

Default value: <u>nop</u>

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

$_{193}$ #$_{194}$ +$_{195}$ K$_{196}$ **$fcol**

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

$fmtlead

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

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 <u>set</u> (<u>^XA</u>) command to set $fmtlead to "**@**".

Default value: empty string

$^{$}_{209}$ #$_{210}$ +$_{211}$ K$_{212}$ **$gflags**

Some of the ways MicroEMACS controls its internal functions can be modified by the value in the $gflags <u>variable</u>. 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
212^{K} $gflags

$_{213}$ #$_{214}$ +$_{215}$ K$_{216}$ **$hardtab**

This <u>variable</u> 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
216^{K} $hardtab

$^{$}_{217}$ #$_{218}$ +$_{219}$ K$_{220}$ **$hjump**

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

Default value: 1

217$^{$}$ $hjump variable
218$^{#}$ _hjump
219^{+} EVariables:hjump
220^{K} $hjump

$^{$}_{221}$ #$_{222}$ +$_{223}$ K$_{224}$ **$hscroll**

This <u>variable</u> 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.

221$^{$}$ $hscroll variable
222$^{#}$ _hscroll
223^{+} EVariables:hscroll
224^{K} $hscroll

$\$_{225}$ #$_{226}$ +$_{227}$ K$_{228}$ **$hscrlbar**

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

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

Default value: TRUE

225$^\$$ $hscrlbar variable
226$^\#$ _hscrlbar
227$^+$ EVariables:hscrlbar
228^K $hscrlbar;scroll bar

$_{229}$ #$_{230}$ +$_{231}$ K$_{232}$ **$kill**

This <u>variable</u> contains the first 127 characters currently in the <u>kill buffer</u>.

Attempts to set this variable are ignored.

229[$] $kill variable
230[#] _kill
231[+] EVariables:kill
232[K] $kill

$_{233}$ #$_{234}$ +$_{235}$ K$_{236}$ **$language**

This <u>variable</u> 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.

233$^$ $language variable
234$^#$ _language
235$^+$ EVariables:language
236^K $language

$^\$_{237}$ #$_{238}$ +$_{239}$ K$_{240}$ **$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.

237$^\$$ $lastkey variable
238$^\#$ _lastkey
239$^+$ EVariables:lastkey
240^K $lastkey

$_{241} #_{242} +_{243} K_{244} **$lastmesg**

This <u>variable</u> contains the text of the last message which MicroEMACS wrote on the <u>message line</u>.

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

241$ $lastmesg variable
242# _lastmesg
243+ EVariables:lastmesg
244K $lastmesg

$_{245} #_{246} +_{247} K_{248}$ **$line**

This <u>variable</u> contains the first 127 characters of the current line. Setting this variable overwrites the contents of the current line.

245$ $line variable
246# _line
247+ EVariables:line
248K $line

$_{249} #_{250} +_{251} K_{252} **$lterm**

This <u>variable</u> 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.

249[$] $lterm variable
250[#] _lterm
251[+] EVariables:lterm
252[K] $lterm

$_{253}$ #$_{254}$ +$_{255}$ K$_{256}$ **$lwidth**

This <u>variable</u> contains the number of characters of the current line.

Attempts to set this variable are ignored.

253$^{\$}$ $lwidth variable
254$^{\#}$ _lwidth
255^{+} EVariables:lwidth
256^{K} $lwidth

$₂₅₇ #₂₅₈ +₂₅₉ K₂₆₀ **$match**

This <u>variable</u> contains the last string matched by a <u>search</u> operation.

Attempts to set this variable are ignored.

$261 #262 +263 K264 **$modeflag**

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

Default value: TRUE

261[$] $modeflag variable
262[#] _modeflag
263[+] EVariables:modeflag
264[K] $modeflag

$msflag

Under some operating systems, this boolean <u>variable</u> 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 <u>screens</u>.

Default value: TRUE

265[$] $msflag variable
266[#] _msflag
267[+] EVariables:msflag
268[K] $msflag

$^{\$}$₂₆₉ #₂₇₀ +₂₇₁ K₂₇₂ **$numwind**

This _variable_ contains the number of _windows_ displayed within the current _screen_.
Attempts to set this variable are ignored.

269$ $numwind variable
270# _numwind
271+ EVariables:numwind
272K $numwind

$_{273}$ #$_{274}$ +$_{275}$ K$_{276}$ **$oldcrypt**

If this boolean <u>variable</u> is TRUE, the <u>CRYPT</u> 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.

$₂₇₇ #₂₇₈ +₂₇₉ K₂₈₀ **$orgrow**

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

Setting this variable is equivalent to invoking the <u>change-screen-row</u> command.

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

Default value: 0

$_{281}$ #$_{282}$ +$_{283}$ K$_{284}$ **$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

$_{285} #_{286} +_{287} K_{288} **$pagelen**

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

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

285$ $pagelen variable
286# _pagelen
287+ EVariables:pagelen
288K $pagelen

$^{$}$₂₈₉ #₂₉₀ +₂₉₁ K₂₉₂ **$palette**

This <u>variable</u> contains a string that is used to control the <u>color</u> 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

289$ $palette variable
290# _palette
291+ EVariables:palette
292K $palette

$_{293}$ #$_{294}$ +$_{295}$ K$_{296}$ **$paralead**

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

Default value: Space and TAB characters

293$ $paralead variable
294# _paralead
295+ EVariables:paralead
296K $paralead;paragraph

$^{\$}$₂₉₇ #₂₉₈ ⁺₂₉₉ K₃₀₀ **$pending**

This boolean <u>variable</u> is TRUE if there are type ahead keystrokes waiting to be processed. Attempts to set this variable are ignored.

$\$_{301}$ $\#_{302}$ $+_{303}$ K_{304} **$popflag**

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

<u>apropos</u>
<u>describe-bindings</u>
<u>describe-functions</u>
<u>describe-variables</u>
<u>list-buffers</u>
<u>list-screens</u>
<u>show-files</u>

Default value: TRUE

301$^\$$ $popflag variable
302$^\#$ _popflag
303$^+$ EVariables:popflag
304^K $popflag

$_{305}$ #$_{306}$ +$_{307}$ K$_{308}$ **$posflag**

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

Default value: TRUE

305$^\$$ $posflag variable
306$^\#$ _posflag
307$^+$ EVariables:posflag
308^K $posflag

$_{309}$ #$_{310}$ +$_{311}$ K$_{312}$ **$progname**

This <u>variable</u> 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.

$_{313} \#_{314} +_{315} K_{316}$ **$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

313$ $readhook variable
314# _readhook
315+ EVariables:readhook
316K $readhook;hook;read

$^{$}_{317}$ #$_{318}$ +$_{319}$ K$_{320}$ **$region**

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

Attempts to set this variable are ignored.

317$^{$}$ $region variable
318$^{#}$ _region
319^{+} EVariables:region
320^{K} $region

$\$_{321}$ $\#_{322}$ $+_{323}$ K_{324} **$replace**

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

$rval

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

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

Attempts to set this variable are ignored.

325[$] $rval variable
326[#] _rval
327[+] EVariables:rval
328[K] $rval

[$]₃₂₉ [#]₃₃₀ ⁺₃₃₁ ^K₃₃₂ **$scrname**

This <u>variable</u> contains the current <u>screen</u>'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 <u>rename-screen</u> command.

329[$] $scrname variable
330[#] _scrname
331⁺ EVariables:scrname
332^K $scrname

$³³³ #₃₃₄ +₃₃₅ K₃₃₆ **$search**

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

$₃₃₇ #₃₃₈ +₃₃₉ K₃₄₀ **$searchpnt**

The value of this <u>variable</u> specifies the positioning of the of the <u>point</u> 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

$341 #342 +343 K344 **$seed**

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

Initial value: 0

341[$] $seed variable
342[#] _seed
343[+] EVariables:seed
344[K] $seed

$softtab

The value of this <u>variable</u> relates to the number of spaces inserted by MicroEMACS when the <u>handle-tab</u> 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 <u>space characters</u> in sufficient number to move the <u>point</u> to the next tab stop.

If $softtab is zero, the handle-tab command inserts <u>true tab characters</u>.

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

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

Default value: 0

345$ $softtab variable
346# _softtab
347+ EVariables:softtab
348K $softtab

$_{349}$ #$_{350}$ +$_{351}$ K$_{352}$ **$sres**

This <u>variable</u> 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.

$^{$}_{353}$ #$_{354}$ +$_{355}$ K$_{356}$ **$ssave**

If this boolean <u>variable</u> 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.

353$^{$}$ $ssave variable
354$^{#}$ _ssave
355^{+} EVariables:ssave
356^{K} $ssave

$_{357}$ #$_{358}$ +$_{359}$ K$_{360}$ **$sscroll**

If this boolean <u>variable</u> is TRUE, MicroEMACS is configured for smooth vertical scrolling: when the cursor moves off the top or bottom of the current <u>window</u>, 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

357$^{\$}$ $sscroll variable
358$^{\#}$ _sscroll
359^{+} EVariables:sscroll
360^{K} $sscroll

$₃₆₁ #₃₆₂ +₃₆₃ K₃₆₄ **$status**

This boolean <u>variable</u> contains the status returned by the last command. This is usually used with the <u>!FORCE</u> 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 <u>macro</u>.

$₃₆₅ #₃₆₆ +₃₆₇ K₃₆₈ **$sterm**

This <u>variable</u> contains the character used to terminate search string inputs.

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

365[$] $sterm variable
366[#] _sterm
367[+] EVariables:sterm
368[K] $sterm;replace;search

$₃₆₉ #₃₇₀ +₃₇₁ K₃₇₂ **$target**

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

369[$] $target variable
370[#] _target
371[+] EVariables:target
372[K] $target

$$_{373}$$ #$$_{374}$$ +$$_{375}$$ K$$_{376}$$ **$time**

This <u>variable</u> 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.

373$$^{\$}$$ $time variable
374$$^{\#}$$ _time
375$$^{+}$$ EVariables:time
376^{K} $time

$^{$}$$_{377}$ #$_{378}$ +$_{379}$ K$_{380}$ **$timeflag**

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

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 <u>$time</u>).

377$^{$}$ $timeflag variable
378$^{#}$ _timeflag
379^{+} EVariables:timeflag
380^{K} $timeflag

$${}^\$_{381} \quad {}^\#_{382} \quad {}^+_{383} \quad {}^K_{384}$$ **$tpause**

This <u>variable</u> 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.

381$ $tpause variable
382# _tpause
383+ EVariables:tpause
384K $tpause

$₃₈₅ #₃₈₆ +₃₈₇ K₃₈₈ **$version**

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

Attempts to set this variable are ignored.

385$ $version variable
386# _version
387+ EVariables:version
388K $version

[$]₃₈₉ [#]₃₉₀ ⁺₃₉₁ K₃₉₂ **$vscrlbar**

This boolean <u>variable</u> 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 <u>screen</u>, allowing you to scroll the text in the current <u>window</u> up and down.

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

Default value: TRUE

$_{393}$ #$_{394}$ +$_{395}$ K$_{396}$ **$wchars**

This <u>variable</u> is used to define what a <u>word</u> 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

393$^{\$}$ $wchars variable
394$^{\#}$ _wchars
395^{+} EVariables:wchars
396^{K} $wchars;word

$wline

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

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

$_{401}$ #$_{402}$ +$_{403}$ K$_{404}$ **$wraphook**

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

Default value: <u>wrap-word</u>

$_{405}$ #$_{406}$ +$_{407}$ K$_{408}$ **$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

$\$_{409}$ #$_{410}$ +$_{411}$ K$_{412}$ **$xpos**

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

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

409$^\$$ $xpos variable
410$^\#$ _xpos
411$^+$ EVariables:xpos
412^K $xpos;mouse

[$]₄₁₃ [#]₄₁₄ ⁺₄₁₅ K₄₁₆ **$yankflag**

This boolean <u>variable</u> controls the placement of the <u>point</u> after a <u>yank</u>, <u>yank-pop</u>, <u>insert-file</u> or <u>insert-clip</u> 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

413[$] $yankflag variable
414[#] _yankflag
415[+] EVariables:yankflag
416[K] $yankflag

$_{417}$ #$_{418}$ +$_{419}$ K$_{420}$ **$ypos**

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

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

Interactive Variables

Interactive <u>variables</u> are actually a method to prompt the user for a string. This is done by using an at sign "@" followed with a string <u>argument</u>. The string is displayed on the <u>message line</u>, 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]:

$_{425}$ #$_{426}$ +$_{427}$ K$_{428}$ **User Variables**

User <u>variables</u> 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 <u>macros</u> 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".

Functions

Functions are part of the MicroEMACS <u>Macro language</u>. They can be used wherever an <u>argument</u> (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:

<u>Boolean functions</u>
<u>Numeric functions</u>
<u>String functions</u>
<u>Miscellaneous functions</u>

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>&right</u>, <u>&slower</u>, <u>&supper</u>, <u>&trim</u>, <u>&upper</u> and <u>&xlate</u>

$_{433}$ #$_{434}$ +$_{435}$ K$_{436}$ **Boolean Functions**

These <u>functions</u> perform operations on boolean <u>arguments</u>:

&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

Numeric Functions

These <u>functions</u> perform operations on numerical <u>arguments</u>:

&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 <u>&ascii</u>
&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*

437$ Numeric Functions
438# NumericFunctions
439+ Functions:numericfunctions
440K
function;&abs;&add;&band;⌐&bor;&bxor;&chr;÷&equal;&greater;&isnum;&less;&mod;&negate;⊂&times

$^\$_{441}$ $^\#_{442}$ $^+_{443}$ $^K_{444}$ **String Functions**

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

&ascii *str*   Returns the ASCII code of the first character in *str*. This function is the opposite of <u>&chr</u>

&cat   *str1*   *str2*   Concatenates the two strings to form one

<u>&indirect</u>   *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

&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*

$_{445}$ #$_{446}$ +$_{447}$ K$_{448}$ **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 <u>path</u> and return its full file specification or an empty string if no such file exists
&group *num*		Return <u>group</u> *num* as set by a <u>MAGIC</u> 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

445$^\$$ Miscellaneous Functions
446$^\#$ MiscellaneousFunctions
447$^+$ Functions:zzzmiscellaneousfunctions
448^K function;&bind;&env;∃&find;&group;&gtc;&gtk

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

        ; 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.

449$ &indirect function
450# .indirect
451+ StringFunctions:indirect
452K &indirect;function

Comments

Within the <u>macro language</u>, 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).