$^{\$}_1$ $^{\#}_2$ $^{+}_3$ $^{K}_4$ **abort-command**

Default binding: <u>^G</u>

This <u>command</u> is used interactively to abort out of any command that is waiting for input.

It can be used within a macro to sound a beep but, unless it is used with the <u>!FORCE</u> directive, it causes the macro to abort.

This command is unaffected by <u>numeric arguments</u>.

1$^{\$}$ abort-command
2$^{\#}$ abort_command
3^{+} Commands:abortcommand
4^{K} abort;abort-command

$5 #6 +7 K8 **add-global-mode**

Default binding: M-M

Syntax:

      add-global-mode   *mode*

or:

      add-global-mode   *color*

This command causes the specified mode to be inherited by future (not yet created) buffers (These global modes can later be revoked by the delete-global-mode command). It can also be used to specify the foreground or background color for future windows.

This command does not modify the modes/colors of the current buffer/window. To do so, use the add-mode command.

This command is unaffected by numeric arguments.

5$ add-global-mode
6# add_global_mode
7+ Commands:addglobalmode
8K mode;color;add-global-mode

$^\$_9$ #$_{10}$ +$_{11}$ K$_{12}$ **add-mode**

Default binding: <u>^XM</u>

Syntax:

      add-mode   *mode*

or:

      add-mode   *color*

This <u>command</u> adds the specified <u>mode</u> to the current <u>buffer</u>. It can also be used to specify the foreground or background <u>color</u> for the current <u>window</u>.

To set the default modes/colors for all future buffers/windows, use the <u>add-global-mode</u> command.

This command is unaffected by <u>numeric arguments</u>.

9[$] add-mode
10[#] add_mode
11[+] Commands:addmode
12[K] mode;color;add-mode

$_{13}$ #$_{14}$ +$_{15}$ K$_{16}$ **append-file**

Default binding: <u>^X^A</u>

Syntax:

> append-file   *file name*

Similar to <u>write-file</u>, this <u>command</u> writes out the current <u>buffer</u> to the named file, but rather than replacing its contents, it appends the buffer to the end of the existing text in the file. This does not change the filename of the current buffer. It is especially handy for building log files.

This command is unaffected by <u>numeric arguments</u>.

13$^\$$ append-file command
14$^\#$ append_file
15$^+$ Commands:appendfile
16^K append;file;append-file

$^{\$}_{17}$ $^{\#}_{18}$ $^{+}_{19}$ $^{K}_{20}$ **apropos**

Default binding: <u>M-A</u>

Syntax:

>   apropos   *string*

This <u>command</u> builds a list of all the MicroEMACS <u>commands</u> and <u>macros</u> whose name contains the specified *string*. The list is stored in a buffer named "**Binding list**" and is displayed either in a <u>popup buffer</u> or in a regular window, depending on the value of the <u>$popflag</u> variable.

Commands are listed first, followed by macros (macro names are enclosed in square brackets "**[**" and "**]**"). For each command or macro listed, the associated <u>bindings</u> are also listed.

This command is unaffected by <u>numeric arguments</u>.

17$^{\$}$ apropos command
18$^{\#}$ apropos
19^{+} Commands:apropos
20^{K} apropos

$^{\$}$[21] $^{\#}$[22] $^{+}$[23] ^{K}[24] **backward-character**

Default bindings: <u>^B</u> and <u>FNB</u> (left arrow)

Syntax:

> *n*   backward-character

This <u>command</u> moves the <u>point</u> backward by *n* characters. If *n* is a negative number, the point is moved forward. If no <u>numeric arguments</u> is specified, the point is moved backward by one character.

Note: end of lines count as one character.

If the move would take the point beyond the boundaries of the buffer, this command fails and the point is left at said boundary.

21$^{\$}$ backward-character
22$^{\#}$ backward_character
23^{+} Commands:backwardcharacter
24^{K} character;position;point;backward-character

$^\$$₂₅ #₂₆ +₂₇ K₂₈ **begin-macro**

Default binding: <u>^X(</u>

This <u>command</u> tells MicroEMACS to begin recording all keystrokes, commands and mouse clicks into the <u>keyboard macro</u>. MicroEMACS stops recording when the <u>end-macro</u> (<u>^X)</u>) command is given.

The recording can be replayed by <u>execute-macro</u> (<u>^XE</u>).

This command is unaffected by <u>numeric arguments</u>.

Note: mouse clicks are recorded with the <u>screen</u> (row/column) position they occurred at.

25$^\$$ begin-macro command
26$^\#$ begin_macro
27$^+$ Commands:beginmacro
28^K begin-macro;macro

$^{\$}$29 #30 +31 K32 **beginning-of-file**

Default binding: <u>M-<</u>

This <u>command</u> causes the <u>point</u> to move to the beginning of the buffer.

It is unaffected by <u>numeric arguments</u>.

29$^{\$}$ beginning-of-file
30$^{\#}$ beginning_of_file
31^{+} Commands:beginningoffile
32^{K} beginning;position;point;beginning-of-file

$^{$}_{33}$ $^{#}_{34}$ $^{+}_{35}$ $^{K}_{36}$ **beginning-of-line**

Default binding: <u>^A</u>

This <u>command</u> causes the <u>point</u> to move to the beginning of the current line.

It is unaffected by <u>numeric arguments</u>.

33$^{$}$ beginning-of-line
34$^{#}$ beginning_of_line
35^{+} Commands:beginningofline
36^{K} position;point;beginning-of-line

$^{\$}$37 #38 +39 K40 **bind-to-key**

Default binding: M-K

Syntax:

      bind-to-key   *command name*   *keystroke*

This command associates a command with a *keystroke*, thus creating a binding. A keystroke can be bound only to one command or macro at a time, so when you rebind it, the previous binding is forgotten. On the other hand, a command can have more than one keystroke bound to it.

The *keystroke* is specified using the keystroke syntax or the mouse syntax.

This command cannot be used to specify the key binding for a macro. That is performed by the macro-to-key command.

This command is unaffected by numeric arguments.

37$^{\$}$ bind-to-key command
38$^{\#}$ bind_to_key
39^{+} Commands:bindtokey
40^{K} bind-to-key;binding

$_{41}$ #$_{42}$ +$_{43}$ K$_{44}$ **bind-to-menu**

No default binding

Syntax:

      bind-to-menu   *command name*   *menu name*

This <u>command</u> is available only under Microsoft Windows. It creates a menu item associated with the specified <u>command</u>. The *menu name* is specified using the <u>menu name syntax</u>.

If the *menu name* designates a menu item that already exists, the command fails.

If the *menu name* specifies menus that do not exist yet, they are created as part of the creation of the menu item.

This command cannot be used to bind a <u>macro</u> to a menu. That is performed by the <u>macro-to-menu</u> command.

This command is unaffected by <u>numeric arguments</u>.

41$^{\$}$ bind-to-menu command
42$^{\#}$ bind_to_menu
43^{+} Commands:bindtomenu
44^{K} bind-to-menu;binding;menu

$^{\$}$[45] #[46] $^{+}$[47] K[48] **buffer-position**

Default binding: <u>^X=</u>

This <u>command</u> displays, on the <u>message line</u>, the position of the <u>point</u> within the current window. It lists:

The line (starting at 1), followed by the total number of lines in the buffer

The column (starting at 0), followed by the length of the current line

The character offset (starting at 0, newlines counting as a single character) from the beginning of the buffer, followed by the total number of character in the buffer

The percentage of text before the point

The hexadecimal value of the current character

This command is unaffected by <u>numeric arguments</u>.

45$^{\$}$ buffer-position
46$^{\#}$ buffer_position
47^{+} Commands:bufferposition
48^{K} position;point;buffer-position

$^{\$}$$_{49}$ #$_{50}$ +$_{51}$ K$_{52}$ **cascade-screens**

No default binding

This <u>command</u> is available only under Microsoft Windows. It causes all non-iconic <u>screens</u> to be rearranged in a cascading scheme. If the current screen is maximized (see <u>maximize-screen</u>) at the time this command is invoked, it is restored to its non-maximized size first.

This command is unaffected by <u>numeric arguments</u>.

49$^{\$}$ cascade-screens
50$^{\#}$ cascade_screens
51^{+} Commands:cascadescreens
52^{K} MDI;screen;cascade-screens

$^\${}_{53}$ $^\#{}_{54}$ $^+{}_{55}$ K$_{56}$ **case-region-lower**

Default binding: <u>^X^L</u>

This <u>command</u> causes all the upper case characters in the <u>region</u> to be changed into their lower case counterpart.

The command fails if the <u>mark</u> is not defined in the current <u>window</u>.

This command is unaffected by <u>numeric arguments</u>.

53$^\$$ case-region-lower
54$^\#$ case_region_lower
55$^+$ Commands:caseregionlower
56^K case;region;case-region-lower

$^{$}_{57}$ #$_{58}$ $^{+}_{59}$ K$_{60}$ **case-region-upper**

Default binding: <u>^X^U</u>

This <u>command</u> causes all the lower case characters in the <u>region</u> to be changed into their upper case counterpart.

The command fails if the <u>mark</u> is not defined in the current <u>window</u>.

This command is unaffected by <u>numeric arguments</u>.

57[$] case-region-upper
58[#] case_region_upper
59[+] Commands:caseregionupper
60[K] case;region;case-region-upper

$^{\$}_{61}$ $^{\#}_{62}$ $^{+}_{63}$ $^{K}_{64}$ **case-word-capitalize**

Default binding: <u>M-C</u>

Syntax:

> *n*   case-word-capitalize

This <u>command</u> capitalizes *n* <u>words</u> after the <u>point</u>: it causes the first character of each word to be forced to upper case and the other characters to be forced to lower case. After the command has executed, the point is located just after the last processed word.

Note that since it starts by capitalizing the first letter after the point, this command would normally be issued with the cursor positioned in front of the first letter of the word you wish to capitalize. If you issue it in the middle of a word, you can end up with some strAnge looking text.

The command fails if the <u>numeric argument</u> is negative or if it goes beyond the end of the buffer. If *n* is null, nothing happens. If the numeric argument is not specified, only one word is affected.

61$^{\$}$ case-word-capitalize
62$^{\#}$ case_word_capitalize
63^{+} Commands:casewordcapitalize
64^{K} case;word;case-word-capitalize

$^{\$}_{65}$ #$_{66}$ $^{+}_{67}$ K$_{68}$ **case-word-lower**

Default binding: <u>M-L</u>

Syntax:

>   *n*   case-word-lower

This <u>command</u> forces to lower case *n* <u>words</u> after the <u>point</u>. After the command has executed, the point is located just after the last processed word.

Note that since it starts by processing the first letter after the point, this command would normally be issued with the cursor positioned in front of the first letter of the word you wish to make lower case.

The command fails if the <u>numeric argument</u> is negative or if it goes beyond the end of the buffer. If *n* is null, nothing happens. If the numeric argument is not specified, only one word is affected.

65$^{\$}$ case-word-lower
66$^{\#}$ case_word_lower
67^{+} Commands:casewordlower
68^{K} case;word;case-word-lower

$^{\$}_{69}$ #$_{70}$ +$_{71}$ K$_{72}$ **case-word-upper**

Default binding: <u>M-U</u>

Syntax:

> *n* case-word-upper

This <u>command</u> forces to upper case *n* <u>words</u> after the <u>point</u>. After the command has executed, the point is located just after the last processed word.

Note that since it starts by processing the first letter after the point, this command would normally be issued with the cursor positioned in front of the first letter of the word you wish to make upper case.

The command fails if the <u>numeric argument</u> is negative or if it goes beyond the end of the buffer. If *n* is null, nothing happens. If the numeric argument is not specified, only one word is affected.

69$^{\$}$ case-word-upper
70$^{\#}$ case_word_upper
71^{+} Commands:casewordupper
72^{K} case;word;case-word-upper

$^\$_{73}$ #$_{74}$ +$_{75}$ K$_{76}$ **change-file-name**

Default binding: <u>^XN</u>

Syntax:

      change-file-name   *file name*

This <u>command</u> lets you change the file name associated with the current <u>buffer</u>. It does not change the buffer name. The disk file is unaffected.

This command is unaffected by <u>numeric arguments</u>.

73$^\$$ change-file-name command
74$^\#$ change_file_name
75$^+$ Commands:changefilename
76^K change-file-name;file;name

$^{\$}$₇₇ #₇₈ +₇₉ K₈₀ **change-screen-column**

No default binding.

Syntax:

> *n*   change-screen-column

This <u>command</u> modifies the offset of the current <u>screen</u>'s left column on the desktop. The <u>numeric argument</u> *n* specifies that offset in number of characters. If *n* is not specified, it is taken as zero.

Using this command is equivalent to setting the <u>$orgcol</u> variable.

If *n* is negative or if it is positive but would cause the right border of the screen to be moved off the desktop, the command fails.

Under Microsoft Windows, this command always resets $orgcol to zero and it has no other effect.

77$ change-screen-column command
78# change_screen_column
79+ Commands:changescreencolumn
80K change-screen-column;screen

$^{\$}$₈₁ #₈₂ ⁺₈₃ K₈₄ **change-screen-row**

No default binding.

Syntax:

       n   change-screen-row

This <u>command</u> modifies the offset of the current <u>screen</u>'s top row on the desktop. The <u>numeric argument</u> *n* specifies that offset in number of characters. If *n* is not specified, it is taken as zero.

Using this command is equivalent to setting the <u>$orgrow</u> variable.

If *n* is negative or if it is positive but would cause the bottom border of the screen to be moved off the desktop, the command fails.

Under Microsoft Windows, this command   always resets $orgrow to zero and it has no other effect.

81$^{\$}$ change-screen-row command
82$^{\#}$ change_screen_row
83^{+} Commands:changescreenrow
84^{K} change-screen-row;screen

$\$_{85}$ #$_{86}$ +$_{87}$ K$_{88}$ **change-screen-size**

No default binding.

Syntax:

> *n*   change-screen-size

This <u>command</u> modifies the height of the current <u>screen</u>, causing it to be *n* lines. If the <u>numeric argument</u> *n* is not specified, it is taken to be the height of the whole desktop.

As the height of the screen changes, the bottom <u>window</u> is resized to fit. If the height is decreased, windows that do not fit any more are eliminated, starting from the bottom one.

Using this command is equivalent to setting the <u>$pagelen</u> variable.

If *n* is lower than 3 or if it is greater than the height of the desktop, the command fails.

Under Microsoft Windows:

> The height of a screen does not include the <u>message line</u>.
>
> If *n* is not specified, the command fails.

85$^\$$ change-screen-size command
86$^\#$ change_screen_size
87$^+$ Commands:changescreensize
88^K change-screen-size;screen

$^{\$}$[89] #[90] +[91] K[92] **change-screen-width**

No default binding.

Syntax:

      n   change-screen-width

This <u>command</u> modifies the width of the current <u>screen</u>, causing it to be *n* characters. If the <u>numeric argument</u> *n* is not specified, it is taken to be the width of the whole desktop.

Using this command is equivalent to setting the <u>$curwidth</u> variable.

If *n* is lower than 10 or if it is greater than the width of the desktop, the command fails.

Under Microsoft Windows, if *n* is not specified, the command fails.

89$^{\$}$ change-screen-width command
90$^{\#}$ change_screen_width
91^{+} Commands:changescreenwidth
92^{K} change-screen-width;screen

$_{93}$ #$_{94}$ +$_{95}$ K$_{96}$ **clear-and-redraw**

Default binding: <u>^L</u>

Syntax:

> clear-and-redraw

or:

> *n*   clear-and-redraw

This <u>command</u> performs two different functions, depending on the way it is invoked:

 wether it is invoked with a   or not:

> If the command is invoked without a <u>numeric argument</u>, it causes all <u>screens</u> to be completely repainted.

> If the command is invoked with a numeric argument, it centers the line containing the <u>point</u> in the current <u>window</u>. The value of the numeric argument is irrelevant.

93$^\$$ clear-and-redraw command
94$^\#$ clear_and_redraw
95$^+$ Commands:clearandredraw
96^K clear-and-redraw;screen;window

$^{\$}$[97] #[98] +[99] K[100] **clear-message-line**

No default binding.

This <u>command</u> erases the text (if any) displayed on the <u>message line</u>.

This command is unaffected by <u>numeric arguments</u>.

97[$] clear-message-line command
98[#] clear_message_line
99[+] Commands:clearmessageline
100[K] clear-message-line

$^{$}$₁₀₁ #₁₀₂ +₁₀₃ K₁₀₄ **clip-region**

Default binding: FN^C (Control+Insert)

This command copies the contents of the current region into the clipboard, overwriting any previous clipboard data.

This command is unaffected by numeric arguments.

101$^{$}$ clip-region command
102$^{\#}$ clip_region
103^{+} Commands:clipregion
104^{K} clip-region;clipboard

$^{\$}$[105] $^{\#}$[106] $^{+}$[107] ^{K}[108] **copy-region**

Default binding: <u>M-W</u>

This <u>command</u> copies the contents of the current <u>region</u> into the <u>kill buffer</u>.

This command is unaffected by <u>numeric arguments</u>.

105$^{\$}$ copy-region command
106$^{\#}$ copy_region
107^{+} Commands:copyregion
108^{K} copy-region;region;kill

$\$_{109}$ #$_{110}$ +$_{111}$ K$_{112}$ **count-words**

Default binding: <u>M-^C</u>

This <u>command</u> displays, on the <u>message line</u>, the number of <u>words</u> in the current <u>region</u>, along with the number of characters, lines and the average number of characters per word.

This command is unaffected by <u>numeric arguments</u>.

109$^{\$}$ count-words command
110$^{\#}$ count_words
111^{+} :countwords
112^{K} count-words;word

$^{\$}$[113] $^{\#}$[114] $^{+}$[115] ^{K}[116] **ctlx-prefix**

Default binding: <u>^X</u>

This <u>command</u> is rarely used for execution in the macro language. Its main purpose is to be mentioned in a <u>bind-to-key</u> command, to redefine the **^X** prefix. For instance, the line:

      bind-to-key   ctlx-prefix   FN1

redefines function key **F1** as the prefix to be used in all keystrokes that begin by "**^X-**". After this, keystrokes such as <u>^X^C</u> would be actually typed by pressing and releasing the **F1** key and then pressing the **Control** key and the **C** key together.

113$^{\$}$ ctlx-prefix command
114$^{\#}$ ctlx_prefix
115^{+} Commands:ctlxprefix
116^{K} ctlx-prefix;control x

$^{\$}$₁₁₇ #₁₁₈ ⁺₁₁₉ K₁₂₀ **cut-region**

Default binding: <u>S-FND</u> (Shift+Delete)

This <u>command</u> deletes the contents of the current <u>region</u> after copying them into the <u>clipboard</u>, overwriting any previous clipboard data.

This command is unaffected by <u>numeric arguments</u>.

117$^{\$}$ cut-region command
118$^{\#}$ cut_region
119^{+} Commands:cutregion
120^{K} cut-region

$_{121}$ #$_{122}$ +$_{123}$ K$_{124}$ **cycle-ring**

Default binding: <u>^XY</u>

Syntax:

       n   cycle-ring

This <u>command</u> causes the <u>kill ring</u> to rotate by *n* positions. For instance, if the contents of the kill ring were K_1, K_2 ... K_{14}, K_{15} and K_{16}, the <u>kill buffer</u> would be K_{16}. After a command:

       2   cycle-ring

the kill buffer would be K_{14} and the kill ring would now be ordered: K_{15}, K_{16}, K_1, K_2 ... K_{14}.

If no <u>numeric arguments</u> is specified, this command does not have any effect.

121$^\$$ cycle-ring command
122$^\#$ cycle_ring
123$^+$ Commands:cyclering
124^K cycle-ring;kill

$^{\$}$₁₂₅ #₁₂₆ +₁₂₇ K₁₂₈ **cycle-screens**

Default binding: <u>A-C</u>

This <u>command</u> takes the rearmost <u>screen</u> (actually, the last screen in the screen list) and moves it to the front.

This command is unaffected by <u>numeric arguments</u>.

125$^{\$}$ cycle-screens command
126$^{\#}$ cycle_screens
127^{+} Commands:cyclescreens
128^{K} cycle-screens;screen

$^{$}_{129}$ $^{#}_{130}$ $^{+}_{131}$ $^{K}_{132}$ **delete-blank-lines**

Default binding: <u>^X^O</u>

If the <u>point</u> is on an empty line, this <u>command</u> deletes all the empty lines around (above and below) the current line. If the point is on a non empty line then this command deletes all of the empty lines immediately following that line.

This command is unaffected by <u>numeric arguments</u>.

129$^{$}$ delete-blank-lines command
130$^{#}$ delete_blank_lines
131^{+} Commands:deleteblanklines
132^{K} delete-blank-lines;delete;line

$^{\$}$₁₃₃ **#**₁₃₄ **+**₁₃₅ K₁₃₆ **delete-buffer**

Default binding: <u>^XK</u>

Syntax:

      delete-buffer   *buffer name*

This <u>command</u> attempts to discard the named <u>buffer</u>, reclaiming the memory it occupied. It will not allow the destruction of a buffer which is currently visible through any <u>window</u> on any <u>screen</u>.

This command is unaffected by <u>numeric arguments</u>.

133$^{\$}$ delete-buffer command
134$^{\#}$ delete_buffer
135^{+} Commands:deletebuffer
136^{K} delete-buffer;delete;buffer

$_{137}$ #$_{138}$ +$_{139}$ K$_{140}$ **delete-global-mode**

Default binding: M-^M

Syntax:

Syntax:

      delete-global-mode   *mode*

or:

      delete-global-mode   *color*

This command causes the specified mode to be removed from the ones inherited by future (not yet created) buffers (such global modes would have been set by the add-global-mode command). It can also be used to specify the foreground or background color for future windows.

This command does not modify the modes/colors of the current buffer/window. To do so, use the delete-mode command.

This command is unaffected by numeric arguments.

137$^\$$ delete-global-mode command
138$^\#$ delete_global_mode
139$^+$ Commands:deleteglobalmode
140^K delete-global-mode;mode;color

$^{$}_{141}$ $^{#}_{142}$ $^{+}_{143}$ $^{K}_{144}$ **delete-kill-ring**

Default binding: <u>M-^Y</u>

This <u>command</u> empties the <u>kill ring</u> (this includes the current contents of the <u>kill buffer</u>) and reclaims the memory space it occupied.

This command is unaffected by <u>numeric arguments</u>.

141$^{$}$ delete-kill-ring command
142$^{#}$ delete_kill_ring
143^{+} Commands:deletekillring
144^{K} delete-kill-ring;kill

$₁₄₅ #₁₄₆ +₁₄₇ K₁₄₈ **delete-mode**

Default binding: <u>^X^M</u>

Syntax:

     delete-mode   *mode*

or:

     delete-mode   *color*

This <u>command</u> removes the specified <u>mode</u> from the current <u>buffer</u> (these modes would have been set by the <u>add-mode</u> or <u>add-global-mode</u> commands). It can also be used to specify the foreground or background <u>color</u> for the current <u>window</u>.

To set the default modes/colors for all future buffers/windows, use the <u>delete-global-mode</u> command.

This command is unaffected by <u>numeric arguments</u>.

145$ delete-mode command
146# delete_mode
147+ Commands:deletemode
148K delete-mode;mode;color

$^\$_{149}$ #$_{150}$ +$_{151}$ K$_{152}$ **delete-next-character**

Default binding: <u>^D</u>

Syntax:

> *n*   delete-next-character

or:

> delete-next-character

If *n* is positive, this <u>command</u> deletes, and stores into the <u>kill buffer</u>, *n* characters after the <u>point</u>. If *n* is negative, the *-n* characters preceding the point are deleted and stored into the kill buffer.

If no <u>numeric argument</u> is specified, the character following the point is deleted, but it is **not stored** into the kill buffer.

If an attempt to delete past the end or beginning of the <u>buffer</u> is made, the command fails.

Note that end of lines are counted as one character each for the purpose of deletion.

149$^\$$ delete-next-character command
150$^\#$ delete_next_character
151$^+$ Commands:deletenextcharacter
152^K delete-next-character;kill

$^{$}_{153}$ #$_{154}$ +$_{155}$ K$_{156}$ **delete-next-word**

Default binding: <u>M-D</u>

Syntax:

> *n*   delete-next-word

This <u>command</u> deletes the text from the <u>point</u> to the beginning of the next <u>word</u>, saving it into the <u>kill buffer</u>.

If a positive <u>numeric argument</u> is present, it specifies the number of words to be deleted. A null numeric argument is treaded as a 1. A negative numeric argument causes the command to fail.

153$^{$}$ delete-next-word command
154$^{#}$ delete_next_word
155^{+} Commands:deletenextword
156^{K} delete-next-word;delete;kill;word

$_{157} #_{158} +_{159} K_{160} **delete-other-windows**

Default binding: ^X1

This command deletes all other windows but the active one from the current screen.   It does not discard or destroy any text, just stops looking at those buffers.

This command is unaffected by numeric arguments.

157$ delete-other-windows command
158# delete_other_windows
159+ Commands:deleteotherwindows
160K delete-other-windows;delete;window

$^{$}_{161}$ #$_{162}$ +$_{163}$ K$_{164}$ **delete-previous-character**

Default binding: <u>^H</u> (Backspace key) and <u>FND</u> (Delete key)

Syntax:

> *n*   delete-previous-character

or:

> delete-previous-character

If *n* is positive, this <u>command</u> deletes, and stores into the <u>kill buffer</u>, the *n* characters preceding the <u>point</u>. If *n* is negative, the *-n* characters following the point are deleted and stored into the kill buffer.

If no <u>numeric argument</u> is specified, the character preceding the point is deleted, but it is **not stored** into the kill buffer.

If an attempt to delete past the end or beginning of the <u>buffer</u> is made, the command fails.

Note that end of lines are counted as one character each for the purpose of deletion.

161$^{$}$ delete-previous-character command
162$^{#}$ delete_previous_character
163^{+} Commands:deletepreviouscharacter
164^{K} delete-previous-character;kill

$_{165}$ #$_{166}$ +$_{167}$ K$_{168}$ **delete-previous-word**

Default binding: <u>M-^H</u>

Syntax:

      n   delete-previous-word

This <u>command</u> deletes the text from the <u>point</u> to the beginning of the previous <u>word</u>, saving it into the <u>kill buffer</u>.

If a positive <u>numeric argument</u> is present, it specifies the number of words to be deleted. A negative or null numeric argument causes the command to fail.

165$^\$$ delete-previous-word command
166$^\#$ delete_previous_word
167$^+$ Commands:deletepreviousword
168^K delete-previous-word;delete;kill;word

$^{$}$₁₆₉ #₁₇₀ +₁₇₁ K₁₇₂ **delete-screen**

Default binding: <u>A-D</u>

Syntax:

delete-screen *screen name*

This <u>command</u> deletes the named <u>screen</u>, providing it is not the active one. Note that <u>buffers</u> being displayed on that screen are not discarded.

This command is unaffected by <u>numeric arguments</u>.

169[$] delete-screen command
170[#] delete_screen
171[+] Commands:deletescreen
172[K] delete-screen;delete;screen

$^{\$}$₁₇₃ #₁₇₄ **⁺₁₇₅** K₁₇₆ **delete-window**

Default binding: <u>^X0</u>

This <u>command</u> removes the active <u>window</u> from the <u>screen</u>, giving its space to the window above (or, if there is none, the window below). It does not discard or destroy any text, just stops looking at that <u>buffer</u>.

If the window is alone on the screen, it cannot be removed and the command fails.

This command is unaffected by <u>numeric arguments</u>.

173$^{\$}$ delete-window command
174$^{\#}$ delete_window
175^{+} Commands:deletewindow
176^{K} delete-window;delete;window

$_{177}$ #$_{178}$ +$_{179}$ K$_{180}$ **describe-bindings**

No default binding

This <u>command</u> creates a list of all <u>commands</u> and <u>macros</u>, each with all the keys which are currently <u>bound</u> to it. Commands are listed first, followed by the macros (macro names are surrounded by square brackets "**[**" and "**]**").

This command is unaffected by <u>numeric arguments</u>.

Note: The list is actually built in a special buffer named "**Binding list**". It is displayed as a <u>popup buffer</u> or in a normal <u>window</u>, depending on the value of the <u>$popflag</u> <u>variable</u>.

177$^\$$ describe-bindings command
178$^\#$ describe_bindings
179$^+$ Commands:describebindings
180^K describe-bindings;binding

$^$_{181}$ #$_{182}$ $^+$_{183}$ K$_{184}$ **describe-functions**

No default binding.

This <u>command</u> creates a list of all the <u>functions</u> available in the MicroEMACS macro language..

This command is unaffected by <u>numeric arguments</u>.

Note: The list is actually built in a special buffer named "**Function list**". It is displayed as a <u>popup buffer</u> or in a normal <u>window</u>, depending on the value of the <u>$popflag</u> <u>variable</u>.

181$^$ describe-functions command
182$^#$ describe_functions
183$^+$ Commands:describefunctions
184^K describe-functions;function

$\$_{185}$ #$_{186}$ +$_{187}$ K$_{188}$ **describe-key**

Default binding: <u>^X?</u>

Syntax:

      describe-key   *keystroke*

This <u>command</u> displays the <u>command</u> or <u>macro</u> bound to the specified *keystroke* on the <u>message line</u> (macro names are surrounded by square brackets "**[**" and "**]**"). If the *keystroke* has no binding, the text "Not Bound" is displayed.

When this command is used within a macro, the *keystroke* is specified using the <u>MicroEMACS keystroke syntax</u> or the <u>mouse syntax</u>(a <u>^G</u>, for instance, is typed as a hat character "**^**" followed by the letter "**G**").

When this command is used interactively mode, it displays a prompt: ": describe-binding" and the keystroke is expected to by typed as if the actual bound command or macro was being invoked (a <u>^G</u>, for instance, is typed by holding down the Control key and pressing the G key).

This command is unaffected by <u>numeric arguments</u>.

185$^\$$ describe-key command
186$^\#$ describe_key
187$^+$ Commands:describekey
188^K describe-key;binding

$\$_{189}$ $\#_{190}$ $+_{191}$ K_{192} **describe-variables**

Default binding:

No default binding.

This underline command creates a list of all the variables and their value. Environmental variables are listed first, followed by user variables.

This command is unaffected by numeric arguments.

Note: The list is actually built in a special buffer named "**Variable list**". It is displayed as a popup buffer or in a normal window, depending on the value of the $popflag variable.

189$^\$$ describe-variables command
190$^\#$ describe_variables
191$^+$ Commands:describevariables
192^K describe-variables;variable

$^{\$}_{193}$ #$_{194}$ +$_{195}$ K$_{196}$ **detab-line** and **detab-region**

Default binding: ^X^D

Syntax:

> *n*   detab-line

or:

> detab-region

These two commands are synonyms. Both cause tab characters to be changed into the appropriate number of spaces in the affected lines (the spacing between tab stops is considered to be the value of the $hardtab variable).

If a numeric arguments is specified, *n* lines, starting from the one containing the point, are affected. If *n* is null, the command modifies no line.

If no numeric argument is specified, all the lines belonging to the current region are affected. If no region is defined, the command modifies no line.

After this command has executed, the point is left at the beginning of the last affected line. The buffer is marked as modified, even if no modification actually took place.

193$^{\$}$ detab-line and detab-region commands
194$^{\#}$ detab_line
195^{+} Commands:detabline
196^{K} detab-line;detab-region;region;tabs

$^{\$}$₁₉₇ #₁₉₈ ⁺₁₉₉ K₂₀₀ **display**

Default binding: <u>^XG</u>

Syntax:

      display   *variable*

This <u>command</u> displays the value of the specified <u>variable</u> on the <u>message line</u>. If *variable* is not an existing <u>environmental variable</u> or <u>user variable</u>, the command fails.

This command is unaffected by <u>numeric arguments</u>.

197$^{\$}$ display command
198$^{\#}$ display
199^{+} Commands:display
200^{K} display;variable

$^{\$}_{201}$ $^{\#}_{202}$ $^{+}_{203}$ $^{K}_{204}$ **end-macro**

Default binding: <u>^X)</u>

This <u>command</u> stops the recording of keystrokes, commands or mouse clicks into the <u>keyboard macro</u>.

The command fails if MicroEMACS is not currently in recording mode.

This command is unaffected by <u>numeric arguments</u>.

See also: <u>begin-macro</u> and <u>execute-macro</u>.

201$^{\$}$ end-macro command
202$^{\#}$ end_macro
203^{+} Commands:endmacro
204^{K} end-macro;macro

$^{\$}_{205}$ $^{\#}_{206}$ $^{+}_{207}$ $^{K}_{208}$ **end-of-file**

Default bindings: <u>M-></u> and <u>FN></u> (End key)

This <u>command</u> places the <u>point</u> at the end of the <u>buffer</u>.

This command is unaffected by <u>numeric arguments</u>.

205$^{\$}$ end-of-file command
206$^{\#}$ end_of_file
207^{+} Commands:endoffile
208^{K} end-of-file

$\$_{209}$ #$_{210}$ +$_{211}$ K$_{212}$ **end-of-line**

Default binding: <u>^E</u>

This <u>command</u> places the <u>point</u> at the end of the current line.

This command is unaffected by <u>numeric arguments</u>.

209$^\$$ end-of-line command
210$^\#$ end_of_line
211$^+$ Commands:endofline
212^K end-of-line

$^{\$}$₂₁₃ #₂₁₄ +₂₁₅ K₂₁₆ **end-of-word**

No default binding.

Syntax:

>   *n*   end-of-word

This <u>command</u> moves the <u>point</u> to the end of the nth following <u>word</u>. If the point was located within a word before invoking the command, that word counts as the first one (thus, if *n* is 1, the point moves to the first character following the current word). If an attempt is made to move past the <u>buffer</u>'s end, the command fails but the point is still moved to the end of the buffer.

If no <u>numeric argument</u> is specified, it is equivalent to *n* = 1.

If *n* is null, the command has no effect.

If *n* is negative, it causes the command to behave like <u>previous-word</u> (invoked with the numeric argument *-n*).

213$^{\$}$ end-of-word command
214$^{\#}$ end_of_word
215^{+} Commands:endofword
216^{K} end-of-word

$^\$_{217}$ $^\#_{218}$ $^+_{219}$ $^K_{220}$ **entab-line** and **entab-region**

Default binding: <u>^X^E</u>

Syntax:

> *n*   entab-line

or:

> entab-region

These two <u>commands</u> are synonyms. Both cause space characters to be compressed into tab characters wherever possible in the affected lines (the spacing between tab stops is considered to be the value of the <u>$hardtab</u> <u>variable</u>).

If a <u>numeric arguments</u> is specified, *n* lines, starting from the one containing the <u>point</u>, are affected. If *n* is null, the command modifies no line.

If no numeric argument is specified, all the lines belonging to the current <u>region</u> are affected. If no region is defined, the command modifies no line.

After this command has executed, the point is left at the beginning of the last affected line. The <u>buffer</u> is marked as modified, even if no modification actually took place.

217[$] entab-line and entab-region commands
218[#] entab_line
219[+] Commands:entabline
220[K] entab-line;entab-region;region;tabs

$^{\$}$[221] $^{\#}$[222] $^{+}$[223] ^{K}[224] **exchange-point-and-mark**

Default binding: ^X^X

Syntax:

      n   exchange-point-and-mark

This command swaps the point and the mark number *n*.

If no numeric argument is specified, it is equivalent to *n* = 0.

If mark*n* does not exist, the command fails.

221[$] exchange-point-and-mark command
222[#] exchange_point_and_mark
223[+] Commands:exchangepointandmark
224[K] exchange-point-and-mark;position;point;mark

$^\$_{225}$ $^\#_{226}$ $^+_{227}$ K_{228} **execute-buffer**

No default binding.

Syntax:

>   *n*   execute-buffer   *buffer*

This <u>command</u> executes the <u>macro language</u> statements from the specified <u>buffer</u>.

The command fails if the *buffer* does not exist or if an executed macro statement (within the *buffer*) fails.

If a positive <u>numeric argument</u> is specified, the buffer is executed *n* times. If *n* is negative or null, the command has no effect.

225$^\$$ execute-buffer command
226$^\#$ execute_buffer
227$^+$ Commands:executebuffer
228^K execute-buffer;macro

$^{\$}_{229}$ $^{\#}_{230}$ $^{+}_{231}$ $^{K}_{232}$ **execute-command-line**

Default binding: <u>M-^X</u>

Syntax:

      execute-command-line   *command line*

This <u>command</u> executes the specified *command line* exactly as if it were part of a <u>macro</u>. This is mostly used interactively to invoke a command but prevent it from fetching its own arguments interactively.

This command is unaffected by <u>numeric arguments</u> (note that the *command line* itself may have its own numeric argument).

229$^{\$}$ execute-command-line command
230$^{\#}$ execute_command_line
231^{+} Commands:executecommandline
232^{K} execute-command-line;command

$_{233}$ #$_{234}$ +$_{235}$ K$_{236}$ **execute-file** or **source**

Default binding: <u>M-^S</u>

Syntax:

> *n*   execute-file   *file*

or:

> *n*   source   *file*

This <u>command</u> executes the <u>macro language</u> statements from the specified *file*, after reading it into an invisible <u>buffer</u>.

The *file* does not need to be a fully qualified path name: if it is a simple filename, it is searched along the <u>path</u>.

The command fails if the *file* cannot be found or if an executed macro statement (within the *file*) fails.

If a positive <u>numeric argument</u> is specified, the *file* is executed *n* times. If *n* is negative or null, the command has no effect.

233$^\$$ execute-file and source commands
234$^\#$ execute_file
235$^+$ Commands:executefile
236^K execute-file;source;macro

$_{237}$ #**238** +**239** K$_{240}$ **execute-macro**

Default binding: ^XE

Syntax:

      n execute-macro

This command replays the last recorded keyboard macro.

If a negative or null numeric argument is specified, the command does nothing. If a positive numeric argument is given, the recorded keyboard macro is played *n* times. If no numeric argument is given, the recorded macro is played once.

The command fails if MicroEMACS is currently in recording mode.

See also: begin-macro and end-macro.

237$^\$$ execute-macro command
238$^\#$ execute_macro
239$^+$ Commands:executemacro
240^K execute-macro;execute;macro

$\$_{241}$ #$_{242}$ +$_{243}$ K$_{244}$ **execute-macro-*n***

Default binding (*n* from 1 to 9): <u>S-FN*n*</u>, for *n* = 10: <u>S-FN0</u>
No default binding for *n* greater than 10.

Syntax:

> *arg*   execute-macro-*n*

MicroEMACS has 40 such <u>commands</u> (i.e. *n* can be a number from 1 to 40). Each causes the execution of the corresponding numbered <u>macro</u> (created by the <u>store-macro</u> command).

If a strictly positive <u>numeric argument</u> is specified, the macro is executed repetitively *arg* times. If *arg* is negative or null, nothing happens.

See also: <u>execute-procedure</u>

241$^\$$ execute-macro-*n* command
242$^\#$ execute_macro_n
243$^+$ Commands:executemacron
244^K execute-macro-n;macro

$^{$}_{245}$ #$_{246}$ +$_{247}$ K$_{248}$ **execute-named-command**

Default binding: <u>M-X</u>

Syntax:

    n execute-named-command   *command*

In interactive mode, this <u>command</u> causes a colon "**:**" to appear on the <u>message line</u>. You can then type the name of the <u>command</u> you want to execute and strike Enter. If you type a space or the <u>meta key</u>, MicroEMACS will attempt to complete the name for you. This interactive use provides access to commands that do not have a key <u>binding</u>.

When used within a <u>macro</u>, **execute-named-command** makes the named *command* behave as if it had been called interactively, thus causing it to prompt the user for any arguments it needs.

If a <u>numeric argument</u> is specified, it is simply transmitted to the named *command*.

245$^{$}$ execute-named-command command
246$^{#}$ execute_named_command
247^{+} Commands:executenamedcommand
248^{K} execute-named-command;command

$_{249}$ #$_{250}$ +$_{251}$ K$_{252}$ **execute-procedure** or **run**

Default binding: M-^E

Syntax:

> *n*   execute-procedure   *macro*

or:

> *n*   run   *macro*

These two commands are synonyms. They both cause the execution of the named *macro* (created by the store-procedure command).

If a strictly positive numeric argument is specified, the *macro* is executed repetitively *n* times. If *n* is negative or null, nothing happens.

See also: execute-macro-*n*

$_{253}$ #$_{254}$ +$_{255}$ K$_{256}$ **execute-program**

Default binding: <u>^X$</u>

Syntax:

>   execute-program   *program*

or:

>   *n*   execute-program   *program*

This <u>command</u> spawns an external *program*, without an intervening shell.

The *program* argument is a string. Note that if it contains spaces (as would be necessary to specify command line options), the string should be quoted.

Under MS-Windows:

>   This command allows you to launch a Windows application from MicroEMACS. The current working directory where the application executes is set to the directory of the file in the current <u>window</u> (or, if that window is not associated to a filename, to the last visited directory).

>   If no <u>numeric argument</u> is specified, MicroEMACS and the launched application run independently. If a numeric argument is specified, MicroEMACS <u>synchronizes</u> with the application.

Note: Under MS-DOS, you cannot use this command to invoke built-in system commands (like DIR, for instance). Use <u>shell-command</u> instead.

253$^{\$}$ execute-program command
254$^{\#}$ execute_program
255^{+} Commands:executeprogram
256^{K} execute-program;execute;spawn

$_{257}$ #$_{258}$ +$_{259}$ K$_{260}$ **exit-emacs**

Default binding: ^X^C

Syntax:

> *n*   exit-emacs

This command terminates MicroEMACS.

If no numeric argument is specified and some buffers contain text that has been changed but not yet saved, you will be asked for a confirmation. If a numeric argument is specified, the command terminates MicroEMACS unconditionally.

257$^\$$ exit-emacs command
258$^\#$ exit_emacs
259$^+$ Commands:exitemacs
260^K exit-emacs;exit;quit

[$]₂₆₁ [#]₂₆₂ ⁺₂₆₃ ^K₂₆₄ **fill-paragraph**

Default binding: <u>M-Q</u>

This <u>command</u> reformats the current <u>paragraph</u>, causing all of its text to be filled out to the current fill column (Which is 72 by default and is set with the <u>set-fill-column</u> command or the <u>$fillcol</u> <u>variable</u>).

This command is unaffected by <u>numeric arguments</u>.

261[$] fill-paragraph command
262[#] fill_paragraph
263⁺ Commands:fillparagraph
264^K fill-paragraph;fill;paragraph

$^\$$₂₆₅ **#₂₆₆** **+₂₆₇** K₂₆₈ **filter-buffer**

Default binding: <u>^X#</u>

Syntax:

   filter-buffer   *program*

This <u>command</u> spawns the external filter *program* (for instance: SORT or FIND) and feeds it the contents of the current <u>buffer</u>. The results replace the original text in the buffer.

Under Microsoft Windows, this command creates a <u>DOS box</u> and <u>synchronizes</u> with it.

This command is unaffected by <u>numeric arguments</u>.

265$^\$$ filter-buffer command
266$^\#$ filter_buffer
267$^+$ Commands:filterbuffer
268^K filter-buffer;filter;buffer;execute;spawn;shell;DOS

$_{269}$ #$_{270}$ +$_{271}$ K$_{272}$ **find-file**

Default binding: <u>^X^F</u>

Syntax:

      find-file   *file name*

If the named file is already loaded somewhere in the editor, this <u>command</u> brings its <u>buffer</u> up in the current <u>window</u>. Otherwise, the file is searched for on disk. If it is found, a new buffer is created and the contents of the file are read into it. If the file does not exist, a new empty buffer is created. In all cases, the buffer is brought up in the current window.

This command is unaffected by <u>numeric arguments</u>.

269$^\$$ find-file command
270$^\#$ find_file
271$^+$ Commands:findfile
272^K find-file;file;open;read

$^\$_{273}$ #$_{274}$ +$_{275}$ K$_{276}$ **find-screen**

Default binding: <u>A-F</u>

Syntax:

  find-screen *screen name*

This <u>command</u> brings up the named <u>screen</u>. If the *screen name* does not exist, a new screen is created. On text systems, this screen is displayed on top of the others. On graphic systems, the OS window containing this screen is brought to the foreground.

This command is unaffected by <u>numeric arguments</u>.

273$^\$$ find-screen command
274$^\#$ find_screen
275$^+$ Commands:findscreen
276^K find-screen;screen

$^{\$}_{277}$ #$_{278}$ +$_{279}$ K$_{280}$ **forward-character**

Default binding: <u>^F</u> and <u>FNF</u> (right arrow)

Syntax:

>   *n*   forward-character

This <u>command</u> moves the <u>point</u> forward by *n* characters. If *n* is a negative number, the point is moved backward. If no <u>numeric arguments</u> is specified, the point is moved forward by one character.

Note: end of lines count as one character.

If the move would take the point beyond the boundaries of the buffer, this command fails and the point is left at said boundary.

277$^{\$}$ forward-character command
278$^{\#}$ forward_character
279^{+} Commands:forwardcharacter
280^{K} character;position;point;forward-character

$[281] #[282] +[283] K[284] **goto-line**

Default binding: <u>M-G</u>

Syntax:

> *n*   goto-line
>
> or
>
> goto-line   *n*

This <u>command</u> moves the <u>point</u> to the first character of line number *n* in the current <u>buffer</u>.

The command fails if *n* is lower than 1 or if the buffer is empty. If *n* is greater than the number of lines in the buffer, the point is simply positioned at the end of the buffer.

281[$] goto-line command
282[#] goto_line
283[+] Commands:gotoline
284[K] position;point;goto-line

$_{285}$ #$_{286}$ +$_{287}$ K$_{288}$ **goto-mark**

Default binding: <u>M-^G</u>

Syntax:

> *n*   goto-mark

This <u>command</u> moves the <u>point</u> to the location of the <u>mark</u> number *n*.

If no <u>numeric arguments</u> is specified, the mark number 0 is used.

If *n* is greater than 9, it is treated as the remainder of the division of *n* by 10.

285$^\$$ goto-mark command
286$^\#$ goto_mark
287$^+$ Commands:gotomark
288^K position;mark;goto-mark

$^{\$}$289 #290 +291 K292 **goto-matching-fence**

Default binding: M-^F

When the point is located on a fence character (curly brace, bracket, or parenthesis), this command will make it jump to the matching fence character.

If the point is not located on a fence character or there is no matching fence, a beep sounds and the command fails.

This command is unaffected by numeric arguments.

289$^{\$}$ goto-matching-fence command
290$^{\#}$ goto_matching_fence
291^{+} Commands:gotomatchingfence
292^{K} goto-matching-fence;fence;brace;parenthesis;bracket

$_{293}$ #$_{294}$ +$_{295}$ K$_{296}$ **grow-window**

Default binding: <u>^X^</u> and <u>^XZ</u>

Syntax:

> *n*   grow-window

If *n* is a positive number, this <u>command</u> increases the height of the current <u>window</u> by *n* lines. The window located immediately below the current window (or, if the current window is at the bottom of the <u>screen</u>, the window located immediately above it) shrinks by *n* lines. If that would cause the shrinking window to become too small to display any text, the command fails.

If the current screen contains only one window, the command fails.

If *n* is a negative number, this command acts as if the <u>shrink-window</u> command had been invoked with the corresponding positive number (-*n*).

If no <u>numeric arguments</u> is specified, the height of the window is increased by one line.

To change the size of the current window by specifying an absolute value, use the <u>resize-window</u> command.

293$^\$$ grow-window command
294$^\#$ grow_window
295$^+$ Commands:growwindow
296^K grow-window;resize;window

$^{\$}$[297] $^{\#}$[298] $^{+}$[299] ^{K}[300] **handle-tab**

Default binding: ^I (Tab key)

Syntax:

> *n*   handle-tab

or:

> handle-tab

The behavior of this command depends on the numeric argument (*n*) that is supplied to it:

> With no argument, it simply inserts a single tab character or enough space characters (depending on its configuration...) to get to the next tab stop.

> With an non-zero argument *n*, tabs stops are reset to every n^{th} column and **handle-tab** is reconfigured to insert space characters in sufficient number to get to the next tab stop. This also sets the $softtab variable to *n*.

> With an argument *n* of zero, **handle-tab** is reconfigured so that it inserts true tab characters (its default behavior) and the tab stop interval is reset to its default value of 8.

The distance which a true tab character moves the cursor is reflected by the value of the $hardtab variable. Initially set to 8, this determines how far each tab stop is placed from the previous one.

297$^{\$}$ handle-tab command
298$^{\#}$ handle_tab
299^{+} Commands:handletab
300^{K} tabs;handle-tab

$^{$}$₃₀₁ #₃₀₂ +₃₀₃ K₃₀₄ **help**

Default binding: <u>M-?</u>

This <u>command</u> brings up a <u>window</u> to display the contents of a text file named EMACS.HLP located on the <u>path</u>. This file usually contains a summary of the MicroEMACS commands and default key bindings.

The command fails if the EMACS.HLP file cannot be found.

This command is unaffected by <u>numeric arguments</u>.

301$^{$}$ help command
302$^{#}$ help
303^{+} Commands:help
304^{K} help

$\$_{305}$ #$_{306}$ +$_{307}$ K$_{308}$ **help-engine**

No default binding.

Syntax:

>   help-engine   *file*   *key*

or:

>   help-engine   *file*

This command invokes the MS Windows WinHelp application to display the specified help *file*. If a *key* is specified, the WinHelp application is instructed to search and display the first topic that matches that *key*. Otherwise, the first topic displayed is the help file's table of content.

This command is unaffected by numeric arguments.

This command is available only under the MS Windows version of MicroEMACS.

305$^\$$ help-engine command
306$^\#$ help_engine
307$^+$ Commands:helpengine
308^K help-engine;help

$^{$}_{309}$ #$_{310}$ +$_{311}$ K$_{312}$ **hunt-backward**

Default binding: <u>A-R</u>

Syntax:

> *n*   hunt-backward

If *n* is a positive number, this <u>command</u> searches backwards for the *n*th occurrence of the search string. That search string is the one that was used the last time a <u>search-forward</u> or <u>search-reverse</u> command was issued. The interpretation of the search string is dependant on whether <u>MAGIC</u> mode is set or not in the current <u>buffer</u>.

If a matching text is found in the <u>buffer</u>, the <u>point</u> is moved to the first character of that text. Otherwise, the command fails. The command also fails if there is no search string.

If *n* is a negative number, this command acts as if the <u>hunt-forward</u> command had been invoked with the corresponding positive number (-*n*).

If no <u>numeric arguments</u> is specified, or if the numeric argument is null, it is equivalent to *n* = 1.

309$^{$}$ hunt-backward command
310$^{#}$ hunt_backward
311^{+} Commands:huntbackward
312^{K} hunt-backward;search

$^\$_{313}$ #$_{314}$ +$_{315}$ K$_{316}$ **hunt-forward**

Default binding: <u>A-S</u>

Syntax:

>   *n*   hunt-forward

If *n* is a positive number, this <u>command</u> searches forward for the *n*th occurrence of the search string. That search string is the one that was used the last time a <u>search-forward</u> or <u>search-reverse</u> command was issued. The interpretation of the search string is dependant on whether <u>MAGIC</u> mode is set or not in the current <u>buffer</u>.

If a matching text is found in the <u>buffer</u>, the <u>point</u> is moved to the first character following that text. Otherwise, the command fails. The command also fails if there is no search string.

If *n* is a negative number, this command acts as if the <u>hunt-backward</u> command had been invoked with the corresponding positive number (-*n*).

If no <u>numeric arguments</u> is specified, or if the numeric argument is null, it is equivalent to *n* = 1.

$\$_{317}$ $\#_{318}$ $+_{319}$ K_{320} **i-shell**

Default binding: <u>^XC</u>

This <u>command</u> spawns a command line shell.

Under MS Windows, this command launches a <u>DOS box</u> (a "shell box" under Windows NT). The current working directory where the shell starts is set to the directory of the file in the current <u>window</u> (or, if that window is not associated to a filename, to the last visited directory).

This command is unaffected by <u>numeric arguments</u>.

317$^\$$ i-shell command
318$^\#$ i_shell
319$^+$ Commands:ishell
320^K i-shell;spawn;DOS;shell

$^{\$}$₃₂₁ #₃₂₂ +₃₂₃ K₃₂₄ **incremental-search**

Default binding: <u>^XS</u>

This <u>command</u> is always interactive. It prompts the user for   a search string but, unlike what happens with the <u>search-forward</u> command, the search happens and the display is updated as each new search character is typed.

While searching towards the end of the <u>buffer</u>, each successive character leaves the <u>point</u> at the end of the entire matched string. Typing a <u>^S</u> causes the next occurrence of the string to be searched for (where the next occurrence does not overlap the current occurrence). A <u>^R</u> changes the direction to a backwards search (as performed by a <u>reverse-incremental-search</u> command), pressing the <u>meta key</u> terminates the search and <u>^G</u> aborts the operation. Pressing the Backspace key (or using <u>^H</u>) backs up to the previous match of the string or, if the starting point is reached, it deletes the last character from the search string.

The characters composing the search string are always interpreted literally. <u>MAGIC</u> mode has no effect on incremental searches.

If the search fails, a beep sounds and the search stalls until the search string is edited back into something that exists (or until the operation is aborted).

This command is unaffected by <u>numeric arguments</u>.

321$^{\$}$ incremental-search command
322$^{\#}$ incremental_search
323^{+} Commands:incrementalsearch
324^{K} incremental-search;search

$^{$}_{325}$ #$_{326}$ $^{+}_{327}$ K$_{328}$ **indent-region**

Default binding: <u>M-)</u>

Syntax:

      n   indent-region

This <u>command</u> inserts *n* tab characters in front of each line within the current <u>region</u>.

If the <u>numeric argument</u> *n* is not specified, one tab is inserted per line.

If <u>CMODE</u> is set in the current <u>buffer</u>, lines that begin by a pound sign "#" are not modified (this is to keep C preprocessor directives flush to the left).

Note: the <u>undent-region</u> command can be used to undo the effect of this command.

325$^{$}$ indent-region command
326$^{#}$ indent_region
327^{+} Commands:indentregion
328^{K} indent-region;tabs;region

$^{\$}_{329}$ #$_{330}$ +$_{331}$ K$_{332}$ **insert-clip**

Default binding: <u>S-FNC</u> (Shift + Insert)

Syntax:

> *n*   insert-clip

This <u>command</u> is only available under MS Windows. It inserts the contents of the Windows clipboard at the <u>point</u>.

If the <u>numeric argument</u> *n* is specified, *n* copies of the clipboard's contents are inserted.

329$^{\$}$ insert-clip command
330$^{\#}$ insert_clip
331^{+} Commands:insertclip
332^{K} insert-clip

$_{333}$ #$_{334}$ +$_{335}$ K$_{336}$ **insert-file**

Default binding: <u>^X^I</u>

Syntax:

      insert-file   *file*

This <u>command</u> inserts the contents of the specified *file* into the current <u>buffer</u>, at the <u>point</u>. After the insertion, the point remains at its original place if the <u>$yankflag</u> <u>variable</u> is TRUE. Otherwise, the point is moved to the end of the inserted text.

This command is unaffected by <u>numeric arguments</u>.

333$^\$$ insert-file command
334$^\#$ insert_file
335$^+$ Commands:insertfile
336^K insert-file;file;read

$^\$_{337}$ #$_{338}$ +$_{339}$ K$_{340}$ **insert-space**

Default binding: <u>^C</u>

Syntax:

      n   insert-space

This <u>command</u> inserts *n* space characters at the <u>point</u>. After the insertion, the point remains at its original place.

If the <u>numeric argument</u> *n* is not specified, a single space character is inserted.

337$^\$$ insert-space command
338$^\#$ insert_space
339$^+$ Commands:insertspace
340^K insert-space

$^{\$}_{341}$ $^{\#}_{342}$ $^{+}_{343}$ $^{K}_{344}$ **insert-string**

No default binding.

Syntax:

  n   insert-string   *string*

This <u>command</u> inserts the specified *string* at the <u>point</u>. After the insertion, the point is moved to the end of the inserted text.

If the <u>numeric argument</u> *n* is specified, *n* copies of the specified *string* are inserted (if *n* is negative, it is taken as -*n*). If *n* is 0, nothing happens.

341$^{\$}$ insert-string command
342$^{\#}$ insert_string
343^{+} Commands:insertstring
344^{K} insert-string

$^$_{345}$ #$_{346}$ +$_{347}$ K$_{348}$ **kill-paragraph**

Default binding: <u>M-^W</u>

Syntax:

      n   kill-paragraph

This <u>command</u> deletes the current <u>paragraph</u>, leaving a copy of it in the <u>kill buffer</u>.

If a positive <u>numeric argument</u> *n* is specified, *n* paragraphs, starting with the current one, are deleted. If *n* is negative or null, nothing happens.

345$^\$$ kill-paragraph command
346$^\#$ kill_paragraph
347$^+$ Commands:killparagraph
348^K kill-paragraph;delete;kill;paragraph

$^{\$}$₃₄₉ $^{\#}$₃₅₀ $^{+}$₃₅₁ ^{K}₃₅₂ **kill-region**

Default binding: <u>^W</u>

This <u>command</u> deletes the characters belonging to the current <u>region</u>, leaving a copy of the deleted text in the <u>kill buffer</u>.

This command is unaffected by <u>numeric arguments</u>.

349$^{\$}$ kill-region command
350$^{\#}$ kill_region
351^{+} Commands:killregion
352^{K} kill-region;kill;region

$^{\$}_{353}$ $^{\#}_{354}$ $^{+}_{355}$ $^{K}_{356}$ **kill-to-end-of-line**

Default binding: <u>^K</u>

Syntax:

> *n*   kill-to-end-of-line

This <u>command's</u> deletes text, leaving a copy of it in the <u>kill buffer</u>. The text affected depends on the <u>numeric arguments</u> applied to the command:

> If it is used without a numeric argument, kill-to-end-of-line truly behaves as its name indicates, deleting the text from the <u>point</u> to the end of the current line, but preserving the newline character, unless the point is located at the end of a line in which case the command just deletes the newline character.

> If the numeric argument is 0, the command deletes the text from the start of the current line up to the point.

> If the numeric argument *n* is positive, the command deletes text from the point forward until *n* newlines have been removed.

> If the numeric argument *n* is negative, the command deletes text from the point backwards until *n* newlines have been removed and the beginning of a line has been reached.

353$^{\$}$ kill-to-end-of-line command
354$^{\#}$ kill_to_end_of_line
355^{+} Commands:killtoendofline
356^{K} kill-to-end-of-line;kill

$_{357} #_{358} +_{359} K_{360}$ **list-buffers**

Default binding: <u>^X^B</u>

Syntax:

>   list-buffers

or:

>   *n*   list-buffers

This <u>command</u> creates a list of all the <u>buffer</u> with, for each buffer, the file it was read from, its size, and the active <u>modes</u>. The list is stored in a buffer named "**[Buffers]**" and is displayed in either a <u>popup buffer</u> or a regular <u>window</u>, depending on the value of the <u>$popflag</u> <u>variable</u>.

Within the list, an at sign "@" in column one shows that a file has already been read into a buffer. A star "*****" in column two means that the contents of the buffer have been modified since the last time they were written to disk. A pound sign "#" in column three indicates the file was to large to read into memory and was truncated.

The <u>modes</u> are shown in columns 5 through 14, using a single letter code for each active mode:

Code	Corresponding mode:
W	<u>WRAP</u>
C	<u>CMODE</u>
E	<u>EXACT</u>
V	<u>VIEW</u>
O	<u>OVER</u>
M	<u>MAGIC</u>
Y	<u>CRYPT</u>
A	<u>ASAVE</u>
R	<u>REP</u>

Used without a <u>numeric argument</u>, list-buffers does not list invisible buffers. If a numeric argument is given, this command lists all buffers, including those hidden buffers used by MicroEMACS for internal data and <u>macros</u> storage.

357$^{\$}$ list-buffers command
358$^{\#}$ list_buffers
359^{+} Commands:listbuffers
360^{K} list-buffers;buffer

$₃₆₁ #₃₆₂ +₃₆₃ K₃₆₄ **list-screens**

Default binding: <u>A-B</u>

This <u>command</u> creates a list of all the <u>screens</u> with, for each screen, the names of the <u>buffers</u> visible in <u>windows</u> on that screen. The list is stored in a buffer named "**[Screens]**" and is displayed in either a <u>popup buffer</u>
or a regular <u>window</u>, depending on the value of the <u>$popflag</u> <u>variable</u>.

This command is unaffected by <u>numeric arguments</u>.

361[$] list-screens command
362[#] list_screens
363⁺ Commands:listscreens
364^K list-screens;screen

$^{\$}$₃₆₅ #₃₆₆ +₃₆₇ K₃₆₈ **macro-to-key**

Default binding: <u>^X^K</u>

Syntax:

    macro-to-key   *macro name   keystroke*

This <u>command</u> associates a <u>macro</u> with a *keystroke*, thus creating a <u>binding</u>. A keystroke can be bound only to one <u>command</u> or macro at a time, so when you rebind it, the previous binding is forgotten. On the other hand, a macro can have more than one keystroke bound to it.

This command cannot be used to specify the key binding for a command. That is performed by the <u>bind-to-key</u> command.

The *keystroke* is specified using the <u>keystroke syntax</u> or the <u>mouse syntax</u>.

This command is unaffected by <u>numeric arguments</u>.

365$^{\$}$ macro-to-key command
366$^{\#}$ macro_to_key
367^{+} Commands:macrotokey
368^{K} macro-to-key;binding

$^{\$}_{369}$ #$_{370}$ $^{+}_{371}$ K$_{372}$ **macro-to-menu**

No default binding

Syntax:

macro-to-menu   *macro name*   *menu name*

This <u>command</u> is available only under Microsoft Windows. It creates a menu item associated with the specified <u>macro</u>. The *menu name* is specified using the <u>menu name syntax</u>.

If the *menu name* designates a menu item that already exists, the command fails.

If the *menu name* specifies menus that do not exist yet, they are created as part of the creation of the menu item.

This command cannot be used to bind a <u>command</u> to a menu. That is performed by the <u>bind-to-menu</u> command.

This command is unaffected by <u>numeric arguments</u>.

369$^{\$}$ macro-to-menu command
370$^{\#}$ macro_to_menu
371^{+} Commands:macrotomenu
372^{K} macro-to-menu;binding;menu

$^\$_{373}$ $^\#_{374}$ $^+_{375}$ $^K_{376}$ **maximize-screen**

No default binding.

This command is available only under Microsoft Windows. It causes the current screen to be enlarged so that it occupies all the available space on MicroEMACS's frame window. If the current screen is already maximized at the time this command is invoked, nothing happens.

This command is unaffected by numeric arguments.

To restore the current screen to the size and position it had before invoking this command, use the restore-screen command.

373$^\$$ maximize-screen command
374$^\#$ maximize_screen
375$^+$ Commands:maximizescreen
376^K MDI;screen;maximize-screen

$_{377}$ #$_{378}$ +$_{379}$ K$_{380}$ **meta-prefix**

Default binding: ^[(Escape key)

This is a dummy command meant to be used in combination with the bind-to-key command in order to redefine the meta key.

For example, to define the F1 function key as being the meta key:

        unbind-key   ^[
        bind-to-key   meta-prefix   FN1

377$ meta-prefix command
378# meta_prefix
379+ Commands:metaprefix
380K meta-prefix

$^{\$}_{381}$ $^{\#}_{382}$ $^{+}_{383}$ $^{K}_{384}$ **minimize-screen**

No default binding.

This <u>command</u> is available only under Microsoft Windows. It causes the current <u>screen</u> to be reduced to an icon. Unless there exists only one screen at the time this command is invoked another screen becomes the current one. If the screen being minimized was maximized (see <u>maximize-screen</u>), the screen becoming current is also maximized.

This command is unaffected by <u>numeric arguments</u>.

To restore the current screen to the size and position it had before invoking this command, use the <u>restore-screen</u> command.

381$^{\$}$ minimize-screen command
382$^{\#}$ minimize_screen
383^{+} Commands:minimizescreen
384^{K} MDI;screen;minimize-screen

$_{385}$ #$_{386}$ +$_{387}$ K$_{388}$ **mouse-move-down**

Default binding: <u>MSa</u> (Press on left mouse button)

This <u>command</u> is meant to be associated with a mouse action. It depends on the <u>$xpos</u> and <u>$ypos</u> <u>variables</u> to contain the coordinates of the mouse pointer. It makes the <u>screen</u> and <u>window</u> where the mouse was clicked the current ones. If the mouse pointer is within the text part of a window (as opposed to the <u>mode line</u>) the point is placed at that position in the text (or at the end of the line if the mouse pointer lies beyond the end of a line).

This command is unaffected by <u>numeric arguments</u>.

Note: Under the MS-Windows version of MicroEMACS, the selection of the current screen is performed by the press on the left mouse button, regardless of the button's binding. Mouse commands themselves cannot select the current screen.

See also: <u>mouse-move-up</u>

385$^{\$}$ mouse-move-down command
386$^{\#}$ mouse_move_down
387^{+} Commands:mousemovedown
388^{K} mouse-move-down

$_{389}$ #$_{390}$ +$_{391}$ K$_{392}$ **mouse-move-up**

Default binding: MSb (Release of left mouse button)

This command is meant to be associated with a mouse action. It depends on the $xpos and $ypos variables to contain the coordinates of the mouse pointer. The actions performed by this command depend of where the previous mouse-move-down command was invoked:

If the mouse pointer was in the mode line part of a window and still is within that mode line, or if it was in the text part of the window and still is, the text in the window is scrolled as if it had been dragged by the mouse. Note that diagonal dragging is possible only if the $diagflag variable is set to TRUE.

If the mouse pointer was on a mode line (except the bottom one), but has moved above or under it, the mode line is moved up or down as if it had been dragged by the mouse, thus resizing the affected windows.

Other cases produce no effect.

The command fails (putting FALSE in the $status variable) if the position of the mouse pointer is the same as that for the last mouse-move-down command. This allows easy detection of lack of mouse movement when the command is used in a macro.

This command is unaffected by numeric arguments.

Note: Under the MS-Windows version of MicroEMACS, the top left and bottom right corners of a screen have no special meaning. Under other versions, mouse-move-up will move the screen if the mouse-move-down was done in the top left corner and resize the screen if mouse-move-down was done in the bottom right corner.

389$^{\$}$ mouse-move-up command
390$^{\#}$ mouse_move_up
391^{+} Commands:mousemoveup
392^{K} mouse-move-up

$\$_{393}$ $\#_{394}$ $+_{395}$ K_{396} **mouse-region-down** and **mouse-region-up**

Default binding: <u>MSe</u> (Press on right mouse button)
      and: <u>MSf</u> (Release of right mouse button)

These <u>commands</u> are meant to be associated with the two parts of a mouse click. Their rather complex behavior is dependant on where the last mouse action took place and is best described by the following topics:

> <u>Copying a Region</u>
> <u>Killing a Region</u>
> <u>Pasting Text</u>

These commands are unaffected by <u>numeric arguments</u>.

393[$] mouse-region-down and mouse-region-up commands
394[#] mouse_region_down
395[+] Commands:mouseregiondown
396[K] mouse-region-down;mouse-region-up

$^{$}_{397}$ #$_{398}$ $^{+}_{399}$ K$_{400}$ **mouse-resize-screen**

No default binding

This <u>command</u> is meant to be associated with a mouse action. It depends on the <u>$xpos</u> and <u>$ypos</u> <u>variables</u> to contain the coordinates of the mouse pointer. It modifies the size of the current <u>screen</u>, bringing its lower right corner to where the mouse was clicked.

This command is unaffected by <u>numeric arguments</u>.

397$^{$}$ mouse-resize-screen command
398$^{#}$ mouse_resize_screen
399^{+} Commands:mouseresizescreen
400^{K} mouse-resize-screen;screen

$^{\$}_{401}$ $^{\#}_{402}$ $^{+}_{403}$ $^{K}_{404}$ **move-window-down**

Default binding: <u>^X^N</u>

Syntax:

> *n*   move-window-down

This <u>command</u> moves the <u>window's</u> view into it's <u>buffer</u> down by *n* lines, causing the text visible in the window to scroll up. If the <u>point</u> scrolls out of view, it is repositioned on the first character of the line located at the center of the window.

If no <u>numeric argument</u> is specified, the text is scrolled by one line.

401$^{\$}$ move-window-down command
402$^{\#}$ move_window_down
403^{+} Commands:movewindowdown
404^{K} move-window-down;scroll;window

$^\$_{405}$ #$_{406}$ +$_{407}$ K$_{408}$ **move-window-up**

Default binding: ^X^P

Syntax:

>   *n*   move-window-up

This command moves the window's view into it's buffer up by *n* lines, causing the text visible in the window to scroll down. If the point scrolls out of view, it is repositioned on the first character of the line located at the center of the window.

If no numeric argument is specified, the text is scrolled by one line.

405$^\$$ move-window-up command
406$^\#$ move_window_up
407$^+$ Commands:movewindowup
408^K move-window-upscroll;window

$^\$_{409}$ $^\#_{410}$ $^+_{411}$ $^K_{412}$ **name-buffer**

Default binding: <u>M-^N</u>

Syntax:

       name-buffer   *name*

This <u>command</u> renames the current <u>buffer</u>, giving it the specified *name*. Note that when a buffer is associated with a file, changing the buffer's name has no effect on the file's name.

If a buffer bearing the specified *name* already exists, another argument is required, and so on until a unique name is supplied.

This command is unaffected by <u>numeric arguments</u>.

409$^\$$ name-buffer command
410$^\#$ name_buffer
411$^+$ Commands:namebuffer
412^K name-buffer;buffer

$^\$_{413}$ #₄₁₄ +₄₁₅ K₄₁₆ **narrow-to-region**

Default binding: ^X<

This <u>command</u> causes the text that does not belong to the current <u>region</u> to become inaccessible until the <u>widen-from-region</u> command is invoked. The <u>mode line</u> displays the symbol "**<>**" to indicate that the current <u>window</u> is associated with a <u>narrowed</u> <u>buffer</u>.

This command is unaffected by <u>numeric arguments</u>.

413$^\$$ narrow-to-region command
414$^\#$ narrow_to_region
415$^+$ Commands:narrowtoregion
416^K narrow-to-region;region;buffer;scope

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

Default binding: <u>^M</u> (Return key)

Syntax:

> *n*   newline

This <u>command</u> inserts *n* newline characters at the <u>point</u>. If the <u>numeric arguments</u> is absent, it is taken as 1.

If *n* is equal to 1 and the <u>buffer</u> is in <u>CMODE</u> mode, C language indentation is performed:

> If the new line is not empty (i.e. the point was not at the end of a line), no other action takes place.

> The new line is indented at the same level as the closest preceding non blank line

> If the newline was inserted right after an opening brace "**{**", the new line is further indented by one tab stop (as if the <u>handle-tab</u> command had been used).

If the buffer is in <u>WRAP</u> mode and the point is past the fill column, wrapping is performed on the last word of the current line before the newline character is inserted.

The command fails if *n* is negative.

417$^\$$ newline command
418$^\#$ newline
419$^+$ Commands:newline
420^K newline

$_{421}$ #$_{422}$ +$_{423}$ K$_{424}$ **newline-and-indent**

Default binding: <u>^J</u>

Syntax:

       n   newline-and-indent

This <u>command</u> inserts *n* newline characters at the <u>point</u>. If the <u>numeric arguments</u> *n* is absent, it is taken as 1.

The new line is indented with enough tab and space characters to match the indentation of the preceding line (the one where the point was when newline-and-indent was invoked).

The command fails if *n* is negative.

421$^\$$ newline-and-indent command
422$^\#$ newline_and_indent
423$^+$ Commands:newlineandindent
424^K newline-and-indent;newline

$\$_{425}$ #$_{426}$ +$_{427}$ K$_{428}$ **next-buffer**

Default binding: <u>^XX</u>

Syntax:

      n   next-buffer

This <u>command</u> causes the current <u>window</u> to display the *n*th next <u>buffer</u> in the circular list of buffers kept by MicroEMACS. If the <u>numeric arguments</u> *n* is absent, it is taken as 1.

The command fails if *n* is not positive.

425$^\$$ next-buffer command
426$^\#$ next_buffer
427$^+$ Commands:nextbuffer
428^K next-buffer;buffer

$^{\$}{}_{429}$ #${}_{430}$ +${}_{431}$ K${}_{432}$ **next-line**

Default binding: <u>^N</u>

Syntax:

> n next-line

This <u>command</u> moves the <u>point</u> to the nth next line. If the <u>numeric arguments</u> n is absent, it is taken as 1.

If n is negative, the point is moved to the nth previous line. If n is 0, nothing happens.

When line move commands (**next-line** or <u>previous-line</u>) are used in a row, the point is kept at the same column it was at before the first of the line moves. If that column lies beyond the end of the current line the point is temporarily brought back to the end of that line.

The command fails if the point is already at the end of the <u>buffer</u> (or the beginning if n is negative).

429$^{\$}$ next-line command
430$^{\#}$ next_line
431^{+} Commands:nextline
432^{K} position;point;next-line

$_{433}$ #$_{434}$ +$_{435}$ K$_{436}$ **next-page**

Default bindings: <u>^V</u> and <u>FNV</u> (Page Down key)

Syntax:

next-page

or:

n   next-page

This <u>command</u> has two different behaviors, depending on the presence or absence of a <u>numeric arguments</u>:

If no numeric argument is specified, the <u>window's</u> view into it's <u>buffer</u> is paged down. If the window contains more than 2 lines of text, the new view overlaps the previous one by two lines: the last two lines of text in the initial view are displayed at the top of the window.

If a positive numeric argument *n* is specified, the window's view into it's buffer is moved down by *n* lines, causing the text visible in the window to scroll up.

If a negative numeric argument *n* is specified, the window's view into it's buffer is moved up by *n* lines, causing the text visible in the window to scroll down, as if the <u>previous-page</u> command had been invoked, with a numeric argument of *-n*.

In all cases, even if a numeric argument of 0 is given, the <u>point</u> is moved to the first character at the top of the window.

433$^{\$}$ next-page command
434$^{\#}$ next_page
435^{+} Commands:nextpage
436^{K} position;next-page

$_{437}$ #$_{438}$ +$_{439}$ K$_{440}$ **next-paragraph**

Default binding: <u>M-N</u>

Syntax:

> *n*   next-paragraph

If used without a <u>numeric arguments</u>, this <u>command</u> moves the <u>point</u> just past the last character of the current <u>paragraph</u> or, if outside a paragraph, to the end of the next paragraph.

If this command is used with a positive numeric argument *n*, the point is moved to the *n*th next end of paragraph.

If *n* is negative, next-paragraph behaves as if the <u>previous-paragraph</u> command had been invoked with an argument of *-n*.

437$^\$$ next-paragraph command
438$^\#$ next_paragraph
439$^+$ Commands:nextparagraph
440^K next-paragraph;position;point;paragraph

$^{\$}_{441}$ #$_{442}$ +$_{443}$ K$_{444}$ **next-window**

Default binding: ^XO

Syntax:

      n   next-window

If used without a <u>numeric arguments</u>, this <u>command</u> makes the next <u>window</u> immediately below the current one the new current window. MicroEMACS updates the highlight of the <u>mode line</u> to indicate the new current window, and places the blinking cursor at the <u>point</u> within that window.

If this command is used with a positive numeric argument *n*, the *n*$^{\text{th}}$ window from the top of the <u>screen</u> is made the current one (window numbering starts at 1).

If *n* is negative, the *-n*$^{\text{th}}$ window from the bottom of the screen is made the current one.

The command fails if *n* (or *-n*) is greater than the number of windows in the screen.

441$^{\$}$ next-window command
442$^{\#}$ next_window
443^{+} Commands:nextwindow
444^{K} next-window;window

$_{445}$ #$_{446}$ +$_{447}$ K$_{448}$ **next-word**

Default bindings: <u>M-F</u> and <u>FN^F</u> (Ctrl + Right arrow)

Syntax:

>   *n*   next-word

This <u>command</u> moves the <u>point</u> to the first character of the *n*th next <u>word</u>. If an attempt is made to move past the <u>buffer</u>'s end, the command fails but the point is still moved to the end of the buffer.

If no <u>numeric argument</u> is specified, it is equivalent to *n* = 1.

If *n* is null, the command has no effect.

If *n* is negative, it causes the command to behave like <u>previous-word</u> (invoked with the numeric argument *-n*).

445$^\$$ next-word command
446$^\#$ next_word
447$^+$ Commands:nextword
448^K next-word;word;position;point

$_{449}$ #$_{450}$ +$_{451}$ K$_{452}$ **nop**

No default binding.

This <u>command</u> has no effect and is unaffected by <u>numeric arguments</u>. Its main purpose is to be the command pointed to by the <u>$bufhook</u>, <u>$cmdhook</u>, <u>$exbhook</u>, <u>$readhook</u> and <u>$writehook</u> <u>variables</u>.

449$^{\$}$ nop command
450$^{\#}$ nop
451^{+} Commands:nop
452^{K} nop

$_{453}$ #$_{454}$ +$_{455}$ K$_{456}$ **open-line**

Default binding: <u>^O</u>

Syntax:

      n   open-line

This <u>command</u> adds *n* newline characters after the <u>point</u>. If the <u>numeric arguments</u> is absent, it is taken as 1.

The command fails if *n* is negative.

453$^\$$ open-line command
454$^\#$ open_line
455$^+$ Commands:openline
456^K open-line;newline

$_{457}$ #$_{458}$ +$_{459}$ K$_{460}$ **overwrite-string**

No default binding.

Syntax:

> overwrite-string   *string*

This <u>command</u> replaces the characters from the <u>point</u> on with the characters from the specified *string*. If the overwriting would extend past the end of the line, the remaining characters from the *string* are simply added at the end of the line (the newline character is not overwritten).

This command is unaffected by <u>numeric arguments</u>.

457$^\$$ overwrite-string command
458$^\#$ overwrite_string
459$^+$ Commands:overwritestring
460^K overwrite-string;OVER

$_{461}$ #$_{462}$ +$_{463}$ K$_{464}$ **pipe-command**

Default binding: <u>^X@</u>

Syntax:

      pipe-command   *program*

This <u>command</u> uses the shell to execute a program, but rather than displaying what the program prints, it attempts to place it in a <u>buffer</u> named "command" to let you edit it and/or save it.

The *program* argument is a string. Note that if it contains spaces (as would be necessary to specify command line options), the string should be quoted.

The <u>VIEW</u> mode is set on the "command" buffer at completion of this command.

Under Microsoft Windows, this command launches the *program* within a <u>DOS box</u> and <u>synchronizes</u> with it. The current working directory where the *program* executes is set to the directory of the file in the current <u>window</u> (or, if that window is not associated to a filename, to the last visited directory).

This command is unaffected by <u>numeric arguments</u>.

461$^{\$}$ pipe-command command
462$^{\#}$ pipe_command
463^{+} Commands:pipecommand
464^{K} pipe-command;execute;spawn;DOS;shell;pipe

$^{\$}_{465}$ $^{\#}_{466}$ $^{+}_{467}$ $^{K}_{468}$ **pop-buffer**

No default binding.

Syntax:

     pop-buffer   *buffer*

or:

     n pop-buffer   *buffer*

This <u>command</u> causes the specified <u>buffer</u> to be displayed as a <u>popup</u> in the current <u>screen</u>.

If a <u>numeric arguments</u> is present, the buffer is marked as being invisible (hidden from the <u>next-buffer</u> command).

465$^{\$}$ pop-buffer command
466$^{\#}$ pop_buffer
467^{+} Commands:popbuffer
468^{K} pop-buffer;buffer

$_{469}$ #$_{470}$ +$_{471}$ K$_{472}$ **previous-line**

Default binding: ^P

Syntax:

    n   previous-line

This <u>command</u> moves the <u>point</u> to the *n*th previous line. If the <u>numeric arguments</u> *n* is absent, it is taken as 1.

If *n* is negative, the point is moved to the *n*th next line. If *n* is 0, nothing happens.

When line move commands (<u>next-line</u> or **previous-line**) are used in a row, the point is kept at the same column it was at before the first of the line moves. If that column lies beyond the end of the current line the point is temporarily brought back to the end of that line.

The command fails if the point is already at the beginning of the <u>buffer</u> (or the end if *n* is negative)

469$^\$$ previous-line command
470$^\#$ previous_line
471$^+$ Commands:previousline
472^K position;point;previous-line

$_{473}$ #$_{474}$ +$_{475}$ K$_{476}$ **previous-page**

Default bindings: <u>M-V</u> and <u>FNZ</u> (Page Up key)

Syntax:

> previous-page

or:

> *n*   previous-page

This <u>command</u> has two different behaviors, depending on the presence or absence of a <u>numeric arguments</u>:

If no numeric argument is specified, the <u>window's</u> view into it's <u>buffer</u> is paged up. If the window contains more than 2 lines of text, the new view overlaps the previous one by two lines: the top two lines of text in the initial view are displayed at the bottom of the window.

If a positive numeric argument *n* is specified, the window's view into it's buffer is moved up by *n* lines, causing the text visible in the window to scroll down.

If a negative numeric argument *n* is specified, the window's view into it's buffer is moved down by *n* lines, causing the text visible in the window to scroll up, as if the <u>next-page</u> command had been invoked, with a numeric argument of -*n*.

In all cases, even if a numeric argument of 0 is given, the <u>point</u> is moved to the first character at the top of the window.

$_{477}$ #$_{478}$ +$_{479}$ K$_{480}$ **previous-paragraph**

Default binding: <u>M-P</u>

Syntax:

> *n*   previous-paragraph

If used without a <u>numeric arguments</u>, this <u>command</u> moves the <u>point</u> to the first character of the current <u>paragraph</u> or, if outside a paragraph, to the beginning of the previous paragraph.

If this command is used with a positive numeric argument *n*, the point is moved back to the *n*th beginning of paragraph.

If *n* is negative, next-paragraph behaves as if the <u>next-paragraph</u> command had been invoked with an argument of *-n*.

477$^\$$ previous-paragraph command
478$^\#$ previous_paragraph
479$^+$ Commands:previousparagraph
480^K previous-paragraph;position;point;paragraph

$_{481}$ #$_{482}$ +$_{483}$ K$_{484}$ **previous-window**

Default binding: <u>^XP</u>

Syntax:

> *n* previous-window

If used without a <u>numeric arguments</u>, this <u>command</u> makes the <u>window</u> immediately above the current one the new current window. MicroEMACS updates the highlight of the <u>mode line</u> to indicate the new current window, and places the blinking cursor at the <u>point</u> within that window.

If this command is used with a positive numeric argument *n*, the *n*$^{\text{th}}$ window from the bottom of the <u>screen</u> is made the current one (window numbering starts at 1).

If *n* is negative, the *-n*$^{\text{th}}$ window from the top of the screen is made the current one.

The command fails if *n* (or *-n*) is greater than the number of windows in the screen.

481$^{\$}$ previous-window command
482$^{\#}$ previous_window
483^{+} Commands:previouswindow
484^{K} previous-window;window

$^\$$485 #486 +487 K488 **previous-word**

Default bindings: M-B and FN^B (Ctrl + Left arrow)

Syntax:

> *n*   previous-word

This command moves the point to the beginning character of the nth preceding word. If the point was located within a word before invoking the command, that word counts as the first one (thus, if *n* is 1, the point moves to the first character of the current word). If an attempt is made to move beyond the buffer's beginning, the command fails but the point is still moved to the beginning of the buffer.

If no numeric argument is specified, it is equivalent to *n* = 1.

If *n* is null, the command has no effect.

If *n* is negative, it causes the command to behave like next-word (invoked with the numeric argument *-n*).

485$^\$$ previous-word command
486$^\#$ previous_word
487$^+$ Commands:previousword
488^K previous-word;position;point;word

$_{489} #_{490} +_{491} K_{492}$ **query-replace-string**

Default binding: <u>M-^R</u>

Syntax:

> *n* query-replace-string *pattern* *replacement*

This <u>command</u> attempts to replace, from the <u>point</u> onward, each piece of text that matches the *pattern* string by the *replacement* string. The *pattern* string is interpreted literally, unless <u>MAGIC</u> mode is enabled in the current <u>buffer</u>.

Each time a match is found, you are queried and can answer by one of the following keystrokes:

Y	replaces the current matching text
N	skips the current match
!	replaces the current matching text and all following matches without anymore queries.
U	jumps back to the last performed replacement and undoes it
<u>^G</u>	aborts the command, leaving the <u>point</u> at its current position
. (dot)	aborts and moves the point back to where the command was originally issued
?	lists the above options

If no <u>numeric arguments</u> is specified, all the matching pieces of text are processed until the end of the buffer is reached. If a positive numeric argument is used, only the first *n* matches are taken into account. If *n* is negative, the command fails.

When this command is invoked interactively, the keystroke used to signal the end of the *pattern* or *replacement* string is specified by the <u>$sterm</u> <u>variable</u> (it is usually the <u>Meta key</u>). Also, both strings may have default values (which are stored in the <u>$search</u> and <u>$replace</u> variables). If you want to replace a string with nothing, and there is a non-empty default for the *replacement* string, striking <u>^K</u> will override that default and enter an empty string instead.

Note: to perform global string replacements without interactive involvement, use the <u>replace-string</u> command.

489$ query-replace-string command
490# query_replace_string
491+ Commands:queryreplacestring
492K query-replace-string;replace

$_{493}$ #$_{494}$ +$_{495}$ K$_{496}$ **quick-exit**

Default binding: M-Z

This command causes MicroEMACS to terminate, but only after having written all the changed buffers into their respective files.

This command is unaffected by numeric arguments.

Note: to terminate MicroEMACS without saving the changed buffers, use the exit-emacs command.

493$^\$$ quick-exit command
494$^\#$ quick_exit
495$^+$ Commands:quickexit
496^K quick-exit;exit;quit

$_{497}$ #$_{498}$ +$_{499}$ K$_{500}$ **quote-character**

Default binding: ^Q

Syntax:

>   *n*   quote

This <u>command</u> inserts literally the next character typed by the user at the <u>point</u>. Even the newline character can be inserted this way, but this causes it to loose its line-splitting meaning.

If a positive <u>numeric arguments</u> is specified, the quoted character is inserted *n* times. If *n* is negative, the command fails. If *n* is null, nothing is inserted, but the typing of a character is still required.

497$^\$$ quote-character command
498$^\#$ quote_character
499$^+$ Commands:quotecharacter
500^K quote-character

$_{501}$ #$_{502}$ +$_{503}$ K$_{504}$ **read-file**

Default binding: <u>^X^R</u>

Syntax:

      read-file   file name

This <u>command</u> reads the named file into the current <u>buffer</u>, replacing the buffer's contents with the text from the file. The file name associated to the buffer is not changed, so you must make sure that replacing the text in the original file with that from the read one is what you are intending when you use this command.

This command is unaffected by <u>numeric arguments</u>.

501$^\$$ read-file command
502$^\#$ read_file
503$^+$ Commands:readfile
504^K read-file;read;file

$^\$$505 #506 +507 K508 **redraw-display**

Default bindings: M-^L and M-!

Syntax:

 n redraw-display

If a non zero numeric argument is specified, this command scrolls the text in the current window so that the current line is displayed as the nth line from the top of the window if *n* is positive, or as the $-n^{th}$ line from the bottom of the window if *n* is negative.

If no numeric argument is specified, or if *n* is zero, the current line is displayed at the center of the window.

505$^\$$ redraw-display command
506$^\#$ redraw_display
507$^+$ Commands:redrawdisplay
508^K redraw-display;window

$_{509}$ #$_{510}$ +$_{511}$ K$_{512}$ **remove-mark**

Default binding: <u>^X__</u> (Ctrl+X Spacebar)

Syntax:

> *n*   remove-mark

This <u>command</u> eliminates the <u>mark</u> number *n*.

If no <u>numeric argument</u> is specified, it is equivalent to *n* = 0.

If mark*n* does not exist, nothing happens.

509$^\$$ remove-mark command
510$^\#$ remove_mark
511$^+$ Commands:removemark
512^K remove-mark;mark

$^\$_{513}$ $^\#_{514}$ $^+_{515}$ $^K_{516}$ **rename-screen**

No default binding.

Syntax:

     rename-screen   *new name*

This <u>command</u> changes the name of the current <u>screen</u> to the specified *new name*. If the *new name* is already in use, the command fails.

This command is unaffected by <u>numeric arguments</u>.

513$^\$$ rename-screen command
514$^\#$ rename_screen
515$^+$ Commands:renamescreen
516^K rename-screen;screen

$^{$}$₅₁₇ #₅₁₈ +₅₁₉ K₅₂₀ **replace-string**

Default binding: <u>M-R</u>

Syntax:

> *n*   replace-string   *pattern   replacement*

This <u>command</u> replaces, from the <u>point</u> onward, each piece of text that matches the *pattern* string by the *replacement* string. The *pattern* string is interpreted literally, unless <u>MAGIC</u> mode is enabled in the current <u>buffer</u>.

If no <u>numeric arguments</u> is specified, all the matching pieces of text are processed until the end of the buffer is reached. If a positive numeric argument is used, only the first *n* matches are processed. If *n* is negative, the command fails.

When this command is used interactively, the keystroke used to signal the end of the *pattern* or *replacement* string is specified by the <u>$sterm</u> <u>variable</u> (it is usually the <u>Meta key</u>). Also, both strings may have default values (which are stored in the <u>$search</u> and <u>$replace</u> variables). If you want to replace a string with nothing, and there is a non-empty default for the *replacement* string, striking <u>^K</u> will override that default and enter an empty string instead.

Note: to have more interactive control over the replacement process, use the <u>query-replace-string</u> command.

517$^{$}$ replace-string command
518$^{#}$ replace_string
519^{+} Commands:replacestring
520^{K} replace-string;replace

$^$_{521} $^#_{522} $^+_{523} $^K_{524}$ **resize-window**

Default binding: <u>^XW</u>

Syntax:

> *n*   resize-window

If *n* is a positive number, this <u>command</u> changes the height of the current <u>window</u> so that it displays *n* lines of text. The window located immediately below the current window (or, if the current window is at the bottom of the <u>screen</u>, the window located immediately above it) shrinks accordingly. If that would cause the shrinking window to become too small to display any text, the command fails.

If the current screen contains only one window, or if *n* is a negative number, the command fails.

If no <u>numeric arguments</u> is specified, nothing happens.

To change the size of the current window by specifying a relative value, use the <u>grow-window</u> or the <u>shrink-window</u> command.

521$^$ resize-window command
522$^#$ resize_window
523$^+$ Commands:resizewindow
524^K resize-window;resize;window

$^{\$}$₅₂₅ #₅₂₆ $^{+}$₅₂₇ K₅₂₈ **restore-screen**

No default binding.

This <u>command</u> is available only under Microsoft Windows. It causes the current <u>screen</u> to be restored to the size and position it had before it was maximized (see <u>maximize-screen</u>) or iconized.(see <u>minimize-screen</u>). If the current screen is neither maximized nor iconized this command has no effect.

This command is unaffected by <u>numeric arguments</u>.

525$^{\$}$ restore-screen command
526$^{\#}$ restore_screen
527^{+} Commands:restorescreen
528^{K} restore-screen;screen

$^$_{529}$ #_{530} $^+_{531}$ K_{532} **restore-window**

No default binding.

This command is only useful when there are multiple windows displayed on the current screen. It causes the window that was current the last time the save-window command was invoked to become the current window again.

If the window that was current the last time **save-window** was invoked no longer exists, or if the screen is not the same, this command fails.

This command is unaffected by numeric arguments.

529$^$ restore-window command
530$^#$ restore_window
531$^+$ Commands:restorewindow
532^K restore-window;save-window

$^\$_{533}$ $^\#_{534}$ $^+_{535}$ $^K_{536}$ **reverse-incremental-search**

Default binding: <u>^XR</u>

This <u>command</u> is always interactive. It prompts the user for   a search string but, unlike what happens with the <u>search-reverse</u> command, the search happens and the display is updated as each new search character is typed.

While searching towards the beginning of the <u>buffer</u>, each successive character leaves the <u>point</u> at the beginning of the matched string. Typing a <u>^R</u> causes the next occurrence of the string to be searched for (where the next occurrence does not overlap the current occurrence). A <u>^S</u> changes the direction to a forward search (as performed by an <u>incremental-search</u> command), pressing the <u>meta key</u> terminates the search and <u>^G</u> aborts the operation. Pressing the Backspace key (or using <u>^H</u>) returns to the previous match of the string or, if the starting point is reached, it deletes the last character from the search string.

The characters composing the search string are always interpreted literally. <u>MAGIC</u> mode has no effect on incremental searches.

If the search fails, a beep sounds and the search stalls until the search string is edited back into something that exists (or until the operation is aborted).

This command is unaffected by <u>numeric arguments</u>.

533$^\$$ reverse-incremental-search command
534$^\#$ reverse_incremental_search
535$^+$ Commands:reverseincrementalsearch
536^K reverse-incremental-search;search

$^{\$}$₅₃₇ #₅₃₈ ⁺₅₃₉ K₅₄₀ **save-file**

Default binding: <u>^X^S</u>

This <u>command</u> writes the contents of the current <u>buffer</u> to disk, if the buffer's contents have been changed since the last read or write operation or the last invocation of the <u>unmark-buffer</u> command.

If the current buffer does not have a file name associated to it (for instance if the buffer has never been subjected to a <u>find-file</u>, <u>read-file</u>, <u>write-file</u> or <u>change-file-name</u> command), the save-file command fails.

If the current buffer is <u>narrowed</u>, a confirmation is requested before writing the text to the file.

This command is unaffected by <u>numeric arguments</u>.

537$^{\$}$ save-file command
538$^{\#}$ save_file
539^{+} Commands:savefile
540^{K} save-file;save;file;write

$^{$}_{541}$ $^{#}_{542}$ $^{+}_{543}$ $^{K}_{544}$ **save-window**

No default binding.

This <u>command</u> saves a reference to the current <u>window,</u> so that the next time the <u>restore-window</u> command is invoked, that window becomes the current window again.

This command is unaffected by <u>numeric arguments</u>.

541$^{$}$ save-window command
542$^{#}$ save_window
543^{+} Commands:savewindow
544^{K} save-window;restore-window

$^{$}$₅₄₅ #₅₄₆ ⁺₅₄₇ K₅₄₈ **scroll-next-down**

Default binding: <u>M-^V</u>

Syntax:

> scroll-next-down

or:

> *n*   scroll-next-down

This <u>command</u> causes the equivalent of a <u>next-page</u> command to be performed on the <u>window</u> located just below the current one (or the top window if the current one is at the bottom of the <u>screen</u>).

If there is only one window displayed in the current screen, this command is equivalent to the next-page command.

545$^{$}$ scroll-next-down command
546$^{#}$ scroll_next_down
547^{+} Commands:scrollnextdown
548^{K} scroll-next-down;next-page

$^\$_{549}$ $^\#_{550}$ $^+_{551}$ $^K_{552}$ **scroll-next-up**

Default binding:

Syntax:

> scroll-next-up

or:

> *n*   scroll-next-up

This <u>command</u> causes the equivalent of a <u>previous-page</u> command to be performed on the <u>window</u> located just below the current one (or the top window if the current one is at the bottom of the <u>screen</u>).

If there is only one window displayed in the current screen, this command is equivalent to the previous-page command.

549$^\$$ scroll-next-up command
550$^\#$ scroll_next_up
551$^+$ Commands:scrollnextup
552^K scroll-next-up;previous-page

$^{$}_{553}$ #$_{554}$ +$_{555}$ K$_{556}$ **search-forward**

Default binding: <u>^S</u>

Syntax:

> *n* search-forward   *search string*

If *n* is a positive number, this <u>command</u> searches forward for the *n*th occurrence of the *search string*. The interpretation of the *search string* is dependant on whether <u>MAGIC</u> mode is set or not in the current <u>buffer</u>.

If a matching text is found in the <u>buffer</u>, the <u>point</u> is moved to the first character following that text. Otherwise, the command fails.

If *n* is a negative number, this command acts as if the <u>search-reverse</u> command had been invoked with the corresponding positive number (-*n*).

If no <u>numeric arguments</u> is specified, or if the numeric argument is null, it is equivalent to *n* = 1.

Note: the *search string* becomes the value of the <u>$search</u> <u>variable</u>

553$^{$}$ search-forward command
554$^{#}$ search_forward
555^{+} Commands:searchforward
556^{K} search-forward;search

$^{\$}$₅₅₇ #₅₅₈ $^{+}$₅₅₉ K₅₆₀ **search-reverse**

Default binding: <u>^R</u>

Syntax:

> *n*   search-reverse   *search string*

If *n* is a positive number, this <u>command</u> searches backwards for the *n*th occurrence of the *search string*. The interpretation of the *search string* is dependant on whether <u>MAGIC</u> mode is set or not in the current <u>buffer</u>.

If a matching text is found in the <u>buffer</u>, the <u>point</u> is moved to the first character of that text. Otherwise, the command fails.

If *n* is a negative number, this command acts as if the <u>search-forward</u> command had been invoked with the corresponding positive number (-*n*).

If no <u>numeric arguments</u> is specified, or if the numeric argument is null, it is equivalent to *n* = 1.

Note: the *search string* becomes the value of the <u>$search</u> <u>variable</u>

557$^{\$}$ search-reverse command
558$^{\#}$ search_reverse
559^{+} Commands:searchreverse
560^{K} search-reverse;search

$_{561}$ #_{562} +_{563} K_{564} **select-buffer**

Default binding: <u>^XB</u>

Syntax:

> select-buffer   *buffer*

or:

> *n*   select-buffer   *buffer*

This <u>command</u> displays the named *buffer* in the current <u>window</u>. If that <u>buffer</u> does not yet exist, it is created.

If a <u>numeric arguments</u> is present, the buffer is marked as being invisible (hidden from the <u>next-buffer</u> command).

561$^\$$ select-buffer command
562$^\#$ select_buffer
563$^+$ Commands:selectbuffer
564^K select-buffer;buffer

$\$_{565}$ $\#_{566}$ $+_{567}$ K_{568} **set**

Default binding: <u>^X^A</u>

Syntax:

> set   *variable*   *value*

or:

> *n*   set   *variable*

This <u>command</u> sets the value of the specified <u>variable</u> to *n* if a <u>numeric arguments</u> is present and to *value* otherwise.

The *variable* must be a <u>user variable</u> or an <u>environmental variable</u> . In the latter case, if the environmental variable does not exist, the command fails.

565$^\$$ set command
566$^\#$ set
567$^+$ Commands:set
568^K set;variable

$^{\$}_{569}$ #$_{570}$ $^{+}_{571}$ K$_{572}$ **set-encryption-key**

Default binding: <u>M-E</u>

Syntax:

      set-encryption-key   *key*

This <u>command</u> sets the current <u>buffer's</u> encryption key (used when the buffer is in <u>CRYPT</u> mode). The specified key can be up to 128 characters long. A length of at least 5 characters is recommended.

This command is unaffected by <u>numeric arguments</u>.

569$^{\$}$ set-encryption-key command
570$^{\#}$ set_encryption_key
571^{+} Commands:setencryptionkey
572^{K} set-encryption-key;encryption

$^{\$}_{573}$ #$_{574}$ +$_{575}$ K$_{576}$ **set-fill-column**

Default binding: ^XF

Syntax:

      n   set-fill-column

This command sets the fill column, (used by the fill-paragraph command) to *n*.

Note that this also sets the $fillcol variable to *n*.

573$^{\$}$ set-fill-column command
574$^{\#}$ set_fill_column
575^{+} Commands:setfillcolumn
576^{K} set-fill-column;fill

$^{$}$~577~ #~578~ +~579~ K~580~ **set-mark**

Default bindings: <u>M-</u> (Ctrl+X Spacebar) and <u>M-.</u>

Syntax:

> *n*   set-mark

This <u>command</u> sets the <u>mark</u> number *n* at the <u>point</u>.

If no <u>numeric argument</u> is specified, it is equivalent to *n* = 0.

577$^{$}$ set-mark command
578$^{#}$ set_mark
579^{+} Commands:setmark
580^{K} set-mark;mark

$_{581}$ #$_{582}$ +$_{583}$ K$_{584}$ **shell-command**

Default binding: <u>^X!</u>

Syntax:

      shell-command   *program*

or:

      n   shell-command   *program*

This <u>command</u> uses the shell to execute the named *program*.

The *program* argument is a string. Note that if it contains spaces (as would be necessary to specify command line options), the string should be quoted.

Under MS-Windows:

> This command launches the *program* within a <u>DOS box</u>. The current working directory where the *program* executes is set to the directory of the file in the current <u>window</u> (or, if that window is not associated to a filename, to the last visited directory).

> If no <u>numeric argument</u> is specified, MicroEMACS and the launched *program* run independently. If a numeric argument is specified, MicroEMACS <u>synchronizes</u> with the *program*.

Note: Under MS-Windows 3.x, you cannot use this command to launch a Windows application. Use <u>execute-program</u> instead.

581$^\$$ shell-command command
582$^\#$ shell_command
583$^+$ Commands:shellcommand
584^K shell-command;execute;spawn;shell;DOS

$^\$_{585}$ #$_{586}$ +$_{587}$ K$_{588}$ **show-files**

No default binding

Syntax:

>show-files   *starname*

This <u>command</u> creates a list of all the files matching the specified *starname*. The starname can contain a directory specification.

For instance, under MS-Windows, the command:

>show-files   "C:\WINDOWS*.INI"

>will create a list of all the files ending by ".INI" in the directory "C:\WINDOWS".

MicroEMACS appends a star "*****" to the end of the specified starname, and appends a dot-star "**.***" if the starname does not contain a dot character. Thus:

>show-files   "C:\WINDOWS\A"

>is equivalent to specifying:

>show-files   "C:\WINDOWS\A*.*"

This command is unaffected by <u>numeric arguments</u>.

Note: The list is actually built in a special buffer named "**File List**". It is displayed as a <u>popup buffer</u> or in a normal <u>window</u>, depending on the value of the <u>$popflag</u> <u>variable</u>.

585$^\$$ show-files command
586$^\#$ show_files
587$^+$ Commands:showfiles
588^K show-files;file

$_{589} #_{590} +_{591} K_{592} **shrink-window**

Default binding: <u>^X^Z</u>

Syntax:

      n   shrink-window

If *n* is a positive number, this <u>command</u> decreases the height of the current <u>window</u> by *n* lines. The window located immediately below the current window (or, if the current window is at the bottom of the <u>screen</u>, the window located immediately above it) grows by *n* lines. If the decrease of height would cause the current window to become too small to display any text, the command fails.

If the current screen contains only one window, the command fails.

If *n* is a negative number, this command acts as if the <u>grow-window</u> command had been invoked with the corresponding positive number (-*n*).

If no <u>numeric arguments</u> is specified, the height of the window is decreased by one line.

To change the size of the current window by specifying an absolute value, use the <u>resize-window</u> command.

589$ shrink-window command
590# shrink_window
591+ Commands:shrinkwindow
592K shrink-window;resize;window

$^{$}_{593}$ #$_{594}$ +$_{595}$ K$_{596}$ **split-current-window**

Default binding: <u>^X2</u>

Syntax:

> *n* split-current-window

This <u>command</u> splits the current <u>window</u> into two windows. Both windows view the current <u>buffer</u> at the current <u>point</u>.

If a <u>numeric arguments</u> is present and not equal to 1, the lower of the two windows becomes current. If *n* = 1, the upper window becomes current.

If no numeric argument is present, the upper window is selected as current if the <u>point</u> was in the upper half of the split window, otherwise, the lower window is selected.

The command fails if it would result in a window too small to display any line of text.

To rid the <u>screen</u> of extraneous windows, use the <u>delete-window</u> or the <u>delete-other-windows</u> commands.

593$^{$}$ split-current-window command
594$^{#}$ split_current_window
595^{+} Commands:splitcurrentwindow
596^{K} split-current-window;window

$\$_{597}$ $\#_{598}$ $+_{599}$ K_{600} **store-macro**

No default binding

Syntax:

> *n*   store-macro
>   *contents*
>   *of*
>   *macro*
> !endm

This <u>command</u> stores the commands and <u>directives</u> that follow it, up to the next <u>!ENDM</u> directive, into a "numbered macro". That <u>macro</u> can be invoked later by the <u>execute-macro-*n*</u> command.

A <u>numeric arguments</u> must be specified and it must be a number from 1 to 40. Otherwise, the command fails.

597$^\$$ store-macro command
598$^\#$ store_macro
599$^+$ Commands:storemacro
600^K store-macro;macro

$^{\$}_{601}$ $^{\#}_{602}$ $^{+}_{603}$ $^{K}_{604}$ **store-procedure**

No default binding

Syntax:

      store-procedure   *name*
         contents
           of
         macro
      !endm

or:

      n   store-procedure
         contents
           of
         macro
      !endm

If no numeric arguments is specified, this command stores the commands and directives that follow it, up to the next !ENDM directive, into a "named macro" or "procedure". That procedure can be invoked later by the run or execute-procedure command, with the argument *name*.

If a numeric argument is specified, this command is equivalent to store-macro.

601$^{\$}$ store-procedure command
602$^{\#}$ store_procedure
603^{+} Commands:storeprocedure
604^{K} store-procedure;macro

$_{605} \#_{606} +_{607} K_{608}$ **tile-screens**

No default binding

Syntax:

      n   tile-screens

This <u>command</u> is available only under Microsoft Windows. It causes all non-iconic <u>screens</u> to be rearranged in a tiled scheme. If the current screen is maximized (see <u>maximize-screen</u>) at the time this command is invoked, it is restored to its non-maximized size first.

If a <u>numeric arguments</u> is present and equals 1, the screens are tiled vertically (i.e. on top of each other). Otherwise, the screens are tiled horizontally (i.e. side by side). However, if there are too many screens to tile (more than 3), the argument is ignored and a mix of vertical and horizontal tiling is used.

605$^\$$ tile-screens command
606$^\#$ tile_screens
607$^+$ Commands:tilescreens
608^K tile-screens;MDI;screen

$_{609}$ #$_{610}$ +$_{611}$ K$_{612}$ **transpose-characters**

Default binding: <u>^T</u>

This <u>command</u> swaps the character that is before the <u>point</u> and the character that is at the point, unless the point is at the end of a line, in which case the two last characters of the line are swapped around.

This command fails if the point is located at the beginning of a line.

This command is unaffected by <u>numeric arguments</u>.

609$^\$$ transpose-characters command
610$^\#$ transpose_characters
611$^+$ Commands:transposecharacters
612^K transpose-characters

$^{\$}_{613}$ #$_{614}$ +$_{615}$ K$_{616}$ **trim-region** or **trim-lines**

Default binding: <u>^X^T</u>

Syntax:

      trim-region

or:

      n   trim-lines

These two <u>command</u> are synonymous. They cause all the trailing space and tab characters between the column position of the <u>point</u> and the end of the processed lines to be deleted.

If a <u>numeric arguments</u> is present, *n* lines, starting from the current one, are processed.

If no numeric argument is present, the lines processed are the ones that belong to the current <u>region</u>.

613$^{\$}$ trim-region and trim-line commands
614$^{\#}$ trim_region
615^{+} Commands:trimregion
616^{K} trim-region;trim-line;region

$_{617}$ #_{618} +_{619} K_{620} **unbind-key**

Default binding: <u>M-^K</u>

Syntax:

      unbind-key   *keystroke*

This <u>command</u> removes the association between a *keystroke* and a <u>macro</u> or a <u>command</u>, thus destroying a <u>binding</u>.

The *keystroke* is specified using the <u>keystroke syntax</u> or the <u>mouse syntax</u>.

This command is unaffected by <u>numeric arguments</u>.

617$ unbind-key command
618# unbind_key
619+ Commands:unbindkey
620K unbind-key;binding

$^{$}_{621}$ $^{\#}_{622}$ $^{+}_{623}$ $^{K}_{624}$ **unbind-menu**

No default binding

Syntax:

      unbind-menu   *menu name*

This <u>command</u> is available only under Microsoft Windows. It destroys a menu item. The *menu name* is specified using the <u>menu name syntax</u>.

If the *menu name* designates a menu item that does not exist, the command fails.

If the *menu name* specifies a menu (that itself contains menu items), all the menu hierarchy under it is destroyed.

This command is unaffected by <u>numeric arguments</u>.

621$^{$}$ unbind-menu command
622$^{\#}$ unbind_menu
623^{+} Commands:unbindmenu
624^{K} unbind-menu;binding;menu

$^\$_{625}$ #$_{626}$ $^+_{627}$ K$_{628}$ **undent-region**

Default binding: <u>M-(</u>

Syntax:

       n   undent-region

This <u>command</u> deletes the first *n* tab characters in front of each line within the current <u>region</u>.

If the <u>numeric argument</u> *n* is not specified, the first tab of each line is deleted.

Note: this command is often used to undo the effect of an <u>indent-region</u> command.

625$^\$$ undent-region command
626$^\#$ undent_region
627$^+$ Commands:undentregion
628^K undent-region;tabs;region

$_{629} #_{630} +_{631} K_{632} **universal-argument**

Default binding: <u>^U</u>

This is a dummy <u>command</u> meant to be used in combination with the <u>bind-to-key</u> command in order to redefine the universal argument key.

To define the F1 function key as being the universal argument key:

    bind-to-key   universal-argument   FN1

Pressing the universal argument key causes a numeric argument of 4 to be generated. If digits (and the minus sign) are entered following the universal argument, they are interpreted to compose a <u>numeric argument</u>, much as if the <u>meta key</u> had been pressed. Also, each further action on the universal argument key multiplies the existing numeric argument by 4.

629[$] universal-argument command
630[#] universal_argument
631[+] Commands:universalargument
632[K] universal-argument;argument

$_{633}$ #$_{634}$ +$_{635}$ K$_{636}$ **unmark-buffer**

Default binding: <u>M-~</u>

This <u>command</u> clears the change flag of the current <u>buffer</u>. This causes MicroEMACS to forget that the buffer's contents have changed since they were last made equivalent to the contents of a disk file (by <u>append-file</u>, <u>find-file</u>, <u>read-file</u>, <u>save-file</u>, <u>view-file</u> or <u>write-file</u>).

This command is unaffected by <u>numeric arguments</u>.

Note: the change flag of the current buffer can also be accessed via the <u>$cbflags</u> <u>variable</u>.

633$^\$$ unmark-buffer command
634$^\#$ unmark_buffer
635$^+$ Commands:unmarkbuffer
636^K unmark-buffer;buffer

$_{637}$ #$_{638}$ +$_{639}$ K$_{640}$ **update-screen**

No default binding

This <u>command</u> immediately updates all elements of the MicroEMACS display during the execution of a <u>macro</u>. It has no visible effect when used interactively.

This command is unaffected by <u>numeric arguments</u>.

637$^\$$ update-screen command
638$^\#$ update_screen
639$^+$ Commands:updatescreen
640^K update-screen;display

$\$_{641}$ #$_{642}$ +$_{643}$ K$_{644}$ **view-file**

Default binding:

Syntax:

     find-file   *file name*

If the named file is already loaded somewhere in the editor, this <u>command</u> brings its <u>buffer</u> up in the current <u>window</u>. Otherwise, the file is searched for on disk. If it is found, a new buffer is created and the contents of the file are read into it. If the file does not exist, a new empty buffer is created. In all cases, the buffer is brought up in the current window, in <u>VIEW</u> mode.

This command is unaffected by <u>numeric arguments</u>.

641$^\$$ view-file command
642$^\#$ view_file
643$^+$ Commands:viewfile
644^K view-file;file;open;read;VIEW

$_{645}$ #$_{646}$ +$_{647}$ K$_{648}$ **widen-from-region**

Default binding: <u>^X></u>

This <u>command</u> causes all the invisible text in the <u>narrowed</u> <u>buffer</u> becomes accessible and visible again.

This command is unaffected by <u>numeric arguments</u>.

645$^\$$ widen-from-region command
646$^\#$ widen_from_region
647$^+$ Commands:widenfromregion
648^K widen-from-region;region;buffer;scope

$^\$_{649}$ $^\#_{650}$ $^+_{651}$ $^K_{652}$ **wrap-word**

No default binding

This command replaces by a newline the first group of space or tab characters preceding the point on the current line. The point is left where it was when the command was invoked.

If no space or tab character is found before the point, a new line is created after the current one and the point is moved to it.

This command is unaffected by numeric arguments.

Note: the $wraphook variable (which points to the command or macro use to perform line wrapping in WRAP mode) is set to wrap-word by default.

649$^\$$ wrap-word command
650$^\#$ wrap_word
651$^+$ Commands:wrapword
652^K wrap-word;word

$^\$_{653}$ $^\#_{654}$ $^+_{655}$ $^K_{656}$ **write-file**

Default binding: <u>^X^W</u>

Syntax:

      write-file   *file name*

This <u>command</u> writes the contents of the current <u>buffer</u> to disk, using the specified *file name*. This file name becomes the one associated with the buffer (indicated by the <u>$cfname</u> <u>variable</u>).

This command is unaffected by <u>numeric arguments</u>.

653$^\$$ write-file command
654$^\#$ write_file
655$^+$ Commands:writefile
656^K write-file;file;write;save

$\$_{657}$ #$_{658}$ $+_{659}$ K$_{660}$ **write-message** or **print**

No default binding

Syntax:

      print   *message*

or:

      write-message   *message*

This <u>command</u> causes the specified *message* to appear on the <u>message line</u>.

This command is unaffected by <u>numeric arguments</u>.

657$^\$$ write-message and print commands
658$^\#$ write_message
659$^+$ Commands:writemessage
660^K write-message;print;message

$_{661} #_{662} +_{663} K_{664} **yank**

Default binding: ^Y

Syntax:

      n yank

This command inserts the contents of the kill buffer at the point. If a numeric arguments is present, the command is repeated *n* times.

If *n* is negative, the command fails.

The placement of the point after the execution of this command is determined by the value of the $yankflag variable.

661[$] yank command
662[#] yank
663[+] Commands:yank
664[K] yank;kill

$₆₆₅ #₆₆₆ +₆₆₇ K₆₆₈ **yank-pop**

Default binding: <u>M-Y</u>

Syntax:

>   *n*   yank-pop

This <u>command</u> cycles the <u>kill ring</u> *n* times (as done by the <u>cycle-ring</u> command) and inserts the contents of the <u>kill buffer</u> at the <u>point</u>. If the previous command was <u>yank</u> or yank-pop, the text inserted by that command is deleted before the new text is inserted.

If no <u>numeric argument</u> is specified, it is equivalent to *n* = 1.

The placement of the point after the execution of this command is determined by the value of the <u>$yankflag</u> <u>variable</u>.

665[$] yank-pop command
666[#] yank_pop
667[+] Commands:yankpop
668[K] yank-pop;yank;kill