$1 #2 +3 K4 abort-command

Default binding: ^G
This command 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 !FORCE directive, it causes the macro to abort.

This command is unaffected by numeric arguments.

1$ abort-command
2# abort_command
3+ Commands:abortcommand
4K 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 K12 add-mode

Default binding: ^XM

Syntax:

```
add-mode   mode
```

or:

```
add-mode   color
```

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

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

This command is unaffected by numeric arguments.

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

$13 #14 +15 K16 append-file

Default binding: ^X^A

Syntax:

```
append-file   file name
```

Similar to write-file, this command writes out the current buffer 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 numeric arguments.

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

$[17] #[18] +[19] K[20] apropos

Default binding: M-A

Syntax:

```
apropos   string
```

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

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

This command is unaffected by numeric arguments.

17$ apropos command
18# apropos
19+ Commands:apropos
20K apropos

$21 #22 +23 K24 backward-character

Default bindings: ^B and FNB (left arrow)

Syntax:

```
n   backward-character
```

This command moves the point backward by *n* characters. If *n* is a negative number, the point is moved forward. If no numeric arguments 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
24K character;position;point;backward-character

$25 #26 +27 K28 begin-macro

Default binding: ^X(

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

The recording can be replayed by execute-macro (^XE).

This command is unaffected by numeric arguments.

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

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

$29 #30 +31 K32 # beginning-of-file

Default binding: M-<

This command causes the point to move to the beginning of the buffer.

It is unaffected by numeric arguments.

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

$33 #34 +35 K36 beginning-of-line

Default binding: ^A

This command causes the point to move to the beginning of the current line.

It is unaffected by numeric arguments.

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

$[37] #[38] +[39] K[40] 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 K44 bind-to-menu

No default binding

Syntax:

```
bind-to-menu   command name   menu name
```

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

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 macro to a menu. That is performed by the macro-to-menu command.

This command is unaffected by numeric arguments.

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

$45 #46 +47 K48 buffer-position

Default binding: ^X=

This command displays, on the message line, the position of the point 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 numeric arguments.

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

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

No default binding

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

This command is unaffected by numeric arguments.

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

$53 #54 +55 K56 case-region-lower

Default binding: ^X^L

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

The command fails if the mark is not defined in the current window.

This command is unaffected by numeric arguments.

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

$57 #58 +59 K60 case-region-upper

Default binding: ^X^U

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

The command fails if the mark is not defined in the current window.

This command is unaffected by numeric arguments.

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

$61 #62 +63 K64 case-word-capitalize

Default binding: M-C

Syntax:

```
n   case-word-capitalize
```

This command capitalizes *n* words after the point: 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 numeric argument 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
64K case;word;case-word-capitalize

$65 #66 +67 K68 case-word-lower

Default binding: M-L

Syntax:

```
n   case-word-lower
```

This command forces to lower case *n* words after the point. 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 numeric argument 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
68K case;word;case-word-lower

$69 #70 +71 K72 case-word-upper

Default binding: M-U

Syntax:

```
n   case-word-upper
```

This command forces to upper case *n* words after the point. 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 numeric argument 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
72K case;word;case-word-upper

$73 #74 +75 K76 change-file-name

Default binding: ^XN

Syntax:

```
change-file-name   file name
```

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

This command is unaffected by numeric arguments.

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

$77 #78 +79 K80 change-screen-column

No default binding.

Syntax:

```
n   change-screen-column
```

This command modifies the offset of the current screen's left column on the desktop. The numeric argument *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 $orgcol 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 command modifies the offset of the current screen's top row on the desktop. The numeric argument *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 $orgrow 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
84K change-screen-row;screen

$[85] #[86] +[87] K[88] change-screen-size

No default binding.

Syntax:

```
n   change-screen-size
```

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

As the height of the screen changes, the bottom window 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 $pagelen 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 message line.

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

Using this command is equivalent to setting the $curwidth 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
92K change-screen-width;screen

$93 #94 +95 K96 clear-and-redraw

Default binding: ^L

Syntax:

```
clear-and-redraw
```

or:

```
n   clear-and-redraw
```

This command 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 numeric argument, it causes all screens to be completely repainted.

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

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

$97 #98 +99 K100 clear-message-line

No default binding.

This command erases the text (if any) displayed on the message line.

This command is unaffected by numeric arguments.

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

$_{101}$ #$_{102}$ +$_{103}$ K$_{104}$ **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
104K clip-region;clipboard

$_{105}$ #$_{106}$ +$_{107}$ K$_{108}$ **copy-region**

Default binding: M-W

This command copies the contents of the current region into the kill buffer.

This command is unaffected by numeric arguments.

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

$109 #110 +111 K112 count-words

Default binding: M-^C

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

This command is unaffected by numeric arguments.

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

$_{113}$ #$_{114}$ +$_{115}$ K$_{116}$ # ctlx-prefix

Default binding: ^X

This command is rarely used for execution in the macro language. Its main purpose is to be mentioned in a bind-to-key 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 ^X^C 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
116K ctlx-prefix;control x

$117 #118 +119 K120 cut-region

Default binding: S-FND (Shift+Delete)

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

This command is unaffected by numeric arguments.

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

$121 #122 +123 K124 cycle-ring

Default binding: ^XY

Syntax:

```
n   cycle-ring
```

This command causes the kill ring 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 kill buffer 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 numeric arguments is specified, this command does not have any effect.

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

$125 #126 +127 K128 cycle-screens

Default binding: A-C

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

This command is unaffected by numeric arguments.

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

$129 #130 +131 K132 # delete-blank-lines

Default binding: ^X^O

If the point is on an empty line, this command 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 numeric arguments.

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

$133 #134 +135 K136 delete-buffer

Default binding: ^XK

Syntax:

```
delete-buffer   buffer name
```

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

This command is unaffected by numeric arguments.

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

$137 #138 +139 K140 delete-global-mode

Default binding: M-^M

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
140K delete-global-mode;mode;color

$141 #142 +143 K144 delete-kill-ring

Default binding: M-^Y

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

This command is unaffected by numeric arguments.

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

$145 #146 +147 K148 delete-mode

Default binding: ^X^M

Syntax:

```
delete-mode   mode
```

or:

```
delete-mode   color
```

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

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

This command is unaffected by numeric arguments.

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

$149 #150 +151 K152 delete-next-character

Default binding: ^D

Syntax:

```
n   delete-next-character
```

or:

```
delete-next-character
```

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

If no numeric argument 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 buffer 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
152K delete-next-character;kill

$153 #154 +155 K156 delete-next-word

Default binding: M-D

Syntax:

```
n   delete-next-word
```

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

If a positive numeric argument 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
156K delete-next-word;delete;kill;word

$157 #158 +159 K160 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 K164 delete-previous-character

Default binding: ^H (Backspace key) and FND (Delete key)

Syntax:

```
n   delete-previous-character
```

or:

```
delete-previous-character
```

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

If no numeric argument 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 buffer 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
164K delete-previous-character;kill

$165 #166 +167 K168 delete-previous-word

Default binding: M-^H

Syntax:

```
n   delete-previous-word
```

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

If a positive numeric argument 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
168K delete-previous-word;delete;kill;word

$169 #170 +171 K172 delete-screen

Default binding: A-D

Syntax:

```
delete-screen   screen name
```

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

This command is unaffected by numeric arguments.

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

$173 #174 +175 K176 delete-window

Default binding: ^X0

This command removes the active window from the screen, 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 buffer.

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

This command is unaffected by numeric arguments.

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

$177 #178 +179 K180 describe-bindings

No default binding

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

This command is unaffected by numeric arguments.

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

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

$181 #182 +183 K184 describe-functions

No default binding.

This command creates a list of all the functions available in the MicroEMACS macro language..

This command is unaffected by numeric arguments.

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

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

$185 #186 +187 K188 describe-key

Default binding: ^X?

Syntax:

```
describe-key   keystroke
```

This command displays the command or macro bound to the specified *keystroke* on the message line (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 MicroEMACS keystroke syntax or the mouse syntax(a ^G, 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 ^G, for instance, is typed by holding down the Control key and pressing the G key).

This command is unaffected by numeric arguments.

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

$_{189}$ #$_{190}$ +$_{191}$ K$_{192}$ # describe-variables

Default binding:

No default binding.

This 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
192K describe-variables;variable

$193 #194 +195 K196 detab-region

Default binding: ^X^D

Syntax:

```
n   detab-region
```

or:

```
detab-region
```

This command causes 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-region command
194# detab_region
195+ Commands:detabregion
196K detab-region;region;tabs

$_{197}$ #$_{198}$ +$_{199}$ K$_{200}$ **display**

Default binding: ^XG

Syntax:

```
display   variable
```

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

This command is unaffected by numeric arguments.

197$ display command
198# display
199+ Commands:display
200K display;variable

$201 #202 +203 K204 end-macro

Default binding: ^X)

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

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

This command is unaffected by numeric arguments.

See also: begin-macro and execute-macro.

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

$205 #206 +207 K208 end-of-file

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

This command places the point at the end of the buffer.

This command is unaffected by numeric arguments.

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

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

Default binding: ^E

This command places the point at the end of the current line.

This command is unaffected by numeric arguments.

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

$213 #214 +215 K216 end-of-word

No default binding.

Syntax:

```
n   end-of-word
```

This command moves the point to the end of the n^{th} following 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 following the current word). If an attempt is made to move past the buffer's end, the command fails but the point is still moved to the end 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 previous-word (invoked with the numeric argument -*n*).

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

$217 #218 +219 K220 entab-region

Default binding: ^X^E

Syntax:

```
n   entab-region
```

or:

```
entab-region
```

This command causes 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 $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.

217$ entab-region command
218# entab_region
219+ Commands:entabregion
220K entab-region;region;tabs

$221 #222 +223 K224 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
224K exchange-point-and-mark;position;point;mark

$225 #226 +227 K228 execute-buffer

No default binding.

Syntax:

```
n   execute-buffer   buffer
```

This command executes the macro language statements from the specified buffer.

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

If a positive numeric argument 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
228K execute-buffer;macro

$229 #230 +231 K232 execute-command-line

Default binding: M-^X

Syntax:

```
execute-command-line   command line
```

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

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

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

$233 #234 +235 K236 **execute-file** or **source**

Default binding: M-^S

Syntax:

```
n   execute-file   file
```

or:

```
n   source   file
```

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

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

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

If a positive numeric argument 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
236K execute-file;source;macro

$237 #238 +239 K240 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
240K execute-macro;execute;macro

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

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

Syntax:

```
arg   execute-macro-n
```

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

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

See also: execute-procedure

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

$245 #246 +247 K248 execute-named-command

Default binding: M-X

Syntax:

```
n   execute-named-command   command
```

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

When used within a macro, **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 numeric argument is specified, it is simply transmitted to the named *command*.

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

$249 #250 +251 K252 **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*

249$ execute-procedure or run commands
250# execute_procedure
251+ Commands:executeprocedure
252K execute-procedure;run;macro

$253 #254 +255 K256 execute-program

Default binding: ^X$

Syntax:

```
execute-program   program
```

or:

```
n   execute-program   program
```

This command 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 window (or, if that window is not associated to a filename, to the last visited directory).

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

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

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

$257 #258 +259 K260 # 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
260K exit-emacs;exit;quit

$261 #262 +263 K264 fill-paragraph

Default binding: M-Q

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

This command is unaffected by numeric arguments.

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

$265 #266 +267 K268 filter-buffer

Default binding: ^X#

Syntax:

```
filter-buffer   program
```

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

Under Microsoft Windows, this command creates a DOS box and synchronizes with it.

This command is unaffected by numeric arguments.

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

$269 #270 +271 K272 # find-file

Default binding: ^X^F

Syntax:

```
find-file   file name
```

If the named file is already loaded somewhere in the editor, this command brings its buffer up in the current window. 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 numeric arguments.

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

$273 #274 +275 K276 find-screen

Default binding: A-F

Syntax:

```
find-screen   screen name
```

This command brings up the named screen. 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 numeric arguments.

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

$277 #278 +279 K280 forward-character

Default binding: ^F and FNF (right arrow)

Syntax:

```
n   forward-character
```

This command moves the point forward by *n* characters. If *n* is a negative number, the point is moved backward. If no numeric arguments 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
280K character;position;point;forward-character

$281 #282 +283 K284 goto-line

Default binding: M-G

Syntax:

```
n   goto-line
or
goto-line   n
```

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

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
284K position;point;goto-line

$285 #286 +287 K288 goto-mark

Default binding: M-^G

Syntax:

```
n   goto-mark
```

This command moves the point to the location of the mark number *n*.

If no numeric arguments 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
288K 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
292K goto-matching-fence;fence;brace;parenthesis;bracket

$293 #294 +295 K296 grow-window

Default binding: ^X^ and ^XZ

Syntax:

```
n   grow-window
```

If *n* is a positive number, this command increases the height of the current window by *n* lines. The window located immediately below the current window (or, if the current window is at the bottom of the screen, 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 shrink-window command had been invoked with the corresponding positive number (-*n*).

If no numeric arguments 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 resize-window command.

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

$297 #298 +299 K300 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
300K tabs;handle-tab

$_{301}$ #$_{302}$ +$_{303}$ K$_{304}$ # help

Default binding: M-?

This command brings up a window to display the contents of a text file named EMACS.HLP located on the path. 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 numeric arguments.

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

$305 #306 +307 K308 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
308K help-engine;help

$309 #310 +311 K312 hunt-backward

Default binding: A-R

Syntax:

```
n   hunt-backward
```

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

If a matching text is found in the buffer, the point 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 hunt-forward command had been invoked with the corresponding positive number (-*n*).

If no numeric arguments 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
312K hunt-backward;search

$313 #314 +315 K316 hunt-forward

Default binding: A-S

Syntax:

```
n   hunt-forward
```

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

If a matching text is found in the buffer, the point 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 hunt-backward command had been invoked with the corresponding positive number (-*n*).

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

313$ hunt-forward command
314# hunt_forward
315+ Commands:huntforward
316K hunt-forward;search

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

Default binding: ^XC

This command spawns a command line shell.

Under MS Windows, this command launches a DOS box (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 window (or, if that window is not associated to a filename, to the last visited directory).

This command is unaffected by numeric arguments.

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

$321 #322 +323 K324 incremental-search

Default binding: ^XS

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

While searching towards the end of the buffer, each successive character leaves the point at the end of the entire matched string. Typing a ^S causes the next occurrence of the string to be searched for (where the next occurrence does not overlap the current occurrence). A ^R changes the direction to a backwards search (as performed by a reverse-incremental-search command), pressing the meta key terminates the search and ^G aborts the operation. Pressing the Backspace key (or using ^H) 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. MAGIC 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 numeric arguments.

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

$325 #326 +327 K328 indent-region

Default binding: M-)

Syntax:

```
n   indent-region
```

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

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

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

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

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

$329 #330 +331 K332 insert-clip

Default binding: S-FNC (Shift + Insert)

Syntax:

```
n   insert-clip
```

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

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

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

$333 #334 +335 K336 insert-file

Default binding: ^X^I

Syntax:

```
insert-file   file
```

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

This command is unaffected by numeric arguments.

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

$337 #338 +339 K340 insert-space

Default binding: ^C

Syntax:

```
n   insert-space
```

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

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

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

$341 #342 +343 K344 insert-string

No default binding.

Syntax:

```
n   insert-string   string
```

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

If the numeric argument *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
344K insert-string

$345 #346 +347 K348 kill-paragraph

Default binding: M-^W

Syntax:

```
n   kill-paragraph
```

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

If a positive numeric argument *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
348K kill-paragraph;delete;kill;paragraph

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

Default binding: ^W

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

This command is unaffected by numeric arguments.

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

$353 #354 +355 K356 kill-to-end-of-line

Default binding: ^K

Syntax:

```
n   kill-to-end-of-line
```

This command's deletes text, leaving a copy of it in the kill buffer. The text affected depends on the numeric arguments 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 point 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
356K kill-to-end-of-line;kill

$357 #358 +359 K360 list-buffers

Default binding: ^X^B

Syntax:

```
list-buffers
```

or:

```
n   list-buffers
```

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

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. A lower than sign "**<**" in column four indicates that the buffer has been narrowed.

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

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

Used without a numeric argument, 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 macros storage.

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

$361 #362 +363 K364 list-screens

Default binding: A-B

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

This command is unaffected by numeric arguments.

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

$365 #366 +367 K368 macro-to-key

Default binding: ^X^K

Syntax:

```
macro-to-key   macro name   keystroke
```

This command associates a macro 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 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 bind-to-key command.

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

This command is unaffected by numeric arguments.

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

$369 #370 +371 K372 macro-to-menu

No default binding

Syntax:

```
macro-to-menu   macro name   menu name
```

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

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 command to a menu. That is performed by the bind-to-menu command.

This command is unaffected by numeric arguments.

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

$373 #374 +375 K376 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
376K MDI;screen;maximize-screen

$377 #378 +379 K380 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 K384 minimize-screen

No default binding.

This command is available only under Microsoft Windows. It causes the current screen 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 maximize-screen), the screen becoming current is also maximized.

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.

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

$385 #386 +387 K388 mouse-move

Default binding: MSm, S-MSm and MS^m (mouse movement)

This command is meant to be associated with a mouse movement. It updates the $xpos and $ypos variables to contain the coordinates of the mouse pointer.

If the $hilight variable is set to a value *n* between 0 and 14, this command updates the mark number *n*+1 so that the highlighted region is automatically updated.

This command is unaffected by numeric arguments.

Note: the mouse actions MSm, S-MSm or MS^m may not be generated for all mouse movements, depending on the value of the $mmove variable.

385$ mouse-move command
386# mouse_move
387+ Commands:mousemove
388K mouse-move

$389 #390 +391 K392 mouse-move-down

Default binding: MSa (Press on 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. It makes the screen and window where the mouse was clicked the current ones. If the mouse pointer is within the text part of a window (as opposed to the mode line) 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 numeric arguments.

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: mouse-move-up

389$ mouse-move-down command
390# mouse_move_down
391+ Commands:mousemovedown
392K mouse-move-down

$393 #394 +395 K396 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.

393$ mouse-move-up command
394# mouse_move_up
395+ Commands:mousemoveup
396K mouse-move-up

$397 #398 +399 K400 **mouse-region-down** and **mouse-region-up**

Default binding: MSe (Press on right mouse button)
and: MSf (Release of right mouse button)

These commands 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:

Copying a Region
Killing a Region
Pasting Text

These commands are unaffected by numeric arguments.

397$ mouse-region-down and mouse-region-up commands
398# mouse_region_down
399+ Commands:mouseregiondown
400K mouse-region-down;mouse-region-up

$401 #402 +403 K404 mouse-resize-screen

No default binding

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. It modifies the size of the current screen, bringing its lower right corner to where the mouse was clicked.

This command is unaffected by numeric arguments.

401$ mouse-resize-screen command
402# mouse_resize_screen
403+ Commands:mouseresizescreen
404K mouse-resize-screen;screen

$405 #406 +407 K408 move-window-down

Default binding: ^X^N

Syntax:

```
n   move-window-down
```

This command moves the window's view into it's buffer down by *n* lines, causing the text visible in the window to scroll up. 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-down command
406# move_window_down
407+ Commands:movewindowdown
408K move-window-down;scroll;window

$409 #410 +411 K412 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.

409$ move-window-up command
410# move_window_up
411+ Commands:movewindowup
412K move-window-upscroll;window

$413 #414 +415 K416 name-buffer

Default binding: M-^N

Syntax:

```
name-buffer   name
```

This command renames the current buffer, 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 numeric arguments.

413$ name-buffer command
414# name_buffer
415+ Commands:namebuffer
416K name-buffer;buffer

$417 #418 +419 K420 narrow-to-region

Default binding: ^X<

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

This command is unaffected by numeric arguments.

417$ narrow-to-region command
418# narrow_to_region
419+ Commands:narrowtoregion
420K narrow-to-region;region;buffer;scope

$421 #422 +423 K424 newline

Default binding: ^M (Return key)

Syntax:

```
n   newline
```

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

If *n* is equal to 1 and the buffer is in CMODE 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 handle-tab command had been used).

If the buffer is in WRAP 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.

421$ newline command
422# newline
423+ Commands:newline
424K newline

$425 #426 +427 K428 newline-and-indent

Default binding: ^J

Syntax:

```
n   newline-and-indent
```

This command inserts *n* newline characters at the point. If the numeric arguments *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.

425$ newline-and-indent command
426# newline_and_indent
427+ Commands:newlineandindent
428K newline-and-indent;newline

$429 #430 +431 K432 # next-buffer

Default binding: ^XX

Syntax:

```
n   next-buffer
```

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

The command fails if *n* is not positive.

429$ next-buffer command
430# next_buffer
431+ Commands:nextbuffer
432K next-buffer;buffer

$[433] #[434] +[435] K[436] next-line

Default binding: ^N

Syntax:

```
n  next-line
```

This command moves the point to the n^{th} next line. If the numeric arguments *n* is absent, it is taken as 1.

If *n* is negative, the point is moved to the n^{th} previous line. If *n* is 0, nothing happens.

When line move commands (**next-line** 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 end of the buffer (or the beginning if *n* is negative).

433[$] next-line command
434[#] next_line
435[+] Commands:nextline
436[K] position;point;next-line

$437 #438 +439 K440 next-page

Default bindings: ^V and FNV (Page Down key)

Syntax:

```
next-page
```

or:

```
n   next-page
```

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

If no numeric argument is specified, the window's view into it's buffer is paged down. The new view overlaps the previous one by the number of lines specified by the $overlap variable. By default, $overlap is equal to 2, so 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 previous-page command had been invoked, with a numeric argument of -*n*.

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

437$ next-page command
438# next_page
439+ Commands:nextpage
440K position;next-page

$441 #442 +443 K444 # next-paragraph

Default binding: M-N

Syntax:

```
n   next-paragraph
```

If used without a numeric arguments, this command moves the point just past the last character of the current paragraph 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 previous-paragraph command had been invoked with an argument of *-n*.

441$ next-paragraph command
442# next_paragraph
443+ Commands:nextparagraph
444K next-paragraph;position;point;paragraph

$445 #446 +447 K448 next-window

Default binding: ^XO

Syntax:

```
n   next-window
```

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

If this command is used with a positive numeric argument *n*, the *n*th window from the top of the screen is made the current one (window numbering starts at 1).

If *n* is negative, the *-n*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.

445$ next-window command
446# next_window
447+ Commands:nextwindow
448K next-window;window

$449 #450 +451 K452 next-word

Default bindings: M-F and FN^F (Ctrl + Right arrow)

Syntax:

```
n   next-word
```

This command moves the point to the first character of the n^{th} next word. If an attempt is made to move past the buffer's end, the command fails but the point is still moved to the end 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 previous-word (invoked with the numeric argument -*n*).

449$ next-word command
450# next_word
451+ Commands:nextword
452K next-word;word;position;point

$453 #454 +455 K456 nop

No default binding.

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

453$ nop command
454# nop
455+ Commands:nop
456K nop

$457 #458 +459 K460 open-line

Default binding: ^O

Syntax:

```
n   open-line
```

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

The command fails if *n* is negative.

457$ open-line command
458# open_line
459+ Commands:openline
460K open-line;newline

$461 #462 +463 K464 overwrite-string

No default binding.

Syntax:

```
overwrite-string   string
```

This command replaces the characters from the point 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 numeric arguments.

461$ overwrite-string command
462# overwrite_string
463+ Commands:overwritestring
464K overwrite-string;OVER

$465 #466 +467 K468 pipe-command

Default binding: ^X@

Syntax:

```
pipe-command   program
```

This command uses the shell to execute a program, but rather than displaying what the program prints, it attempts to place it in a buffer 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 VIEW mode is set on the "command" buffer at completion of this command.

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

This command is unaffected by numeric arguments.

465$ pipe-command command
466# pipe_command
467+ Commands:pipecommand
468K pipe-command;execute;spawn;DOS;shell;pipe

$469 #470 +471 K472 pop-buffer

No default binding.

Syntax:

```
pop-buffer   buffer
```

or:

```
n   pop-buffer   buffer
```

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

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

469$ pop-buffer command
470# pop_buffer
471+ Commands:popbuffer
472K pop-buffer;buffer

$473 #474 +475 K476 previous-line

Default binding: ^P

Syntax:

```
n  previous-line
```

This command moves the point to the n^{th} previous line. If the numeric arguments *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 (next-line 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 buffer (or the end if *n* is negative)

473$ previous-line command
474# previous_line
475+ Commands:previousline
476K position;point;previous-line

$477 #478 +479 K480 previous-page

Default bindings: M-V and FNZ (Page Up key)

Syntax:

```
previous-page
```

or:

```
n  previous-page
```

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

If no numeric argument is specified, the window's view into it's buffer is paged up. The new view overlaps the previous one by the number of lines specified by the $overlap variable. By default, $overlap is equal to 2, so 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 next-page command had been invoked, with a numeric argument of -*n*.

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

477$ previous-page command
478# previous_page
479+ Commands:previouspage
480K position;previous-page

$481 #482 +483 K484 previous-paragraph

Default binding: M-P

Syntax:

```
n  previous-paragraph
```

If used without a numeric arguments, this command moves the point to the first character of the current paragraph 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 next-paragraph command had been invoked with an argument of *-n*.

481$ previous-paragraph command
482# previous_paragraph
483+ Commands:previousparagraph
484K previous-paragraph;position;point;paragraph

$485 #486 +487 K488 previous-window

Default binding: ^XP

Syntax:

```
n   previous-window
```

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

If this command is used with a positive numeric argument *n*, the *n*th window from the bottom of the screen is made the current one (window numbering starts at 1).

If *n* is negative, the *-n*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.

485$ previous-window command
486# previous_window
487+ Commands:previouswindow
488K previous-window;window

$489 #490 +491 K492 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 *n*th 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*).

489$ previous-word command
490# previous_word
491+ Commands:previousword
492K previous-word;position;point;word

$493 #494 +495 K496 query-replace-string

Default binding: M-^R

Syntax:

```
n  query-replace-string  pattern  replacement
```

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

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
^G	aborts the command, leaving the point at its current position
. (dot)	aborts and moves the point back to where the command was originally issued
?	lists the above options

If no numeric arguments 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 $sterm variable (it is usually the Meta key). Also, both strings may have default values (which are stored in the $search and $replace variables). If you want to replace a string with nothing, and there is a non-empty default for the *replacement* string, striking ^K will override that default and enter an empty string instead.

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

493$ query-replace-string command
494# query_replace_string
495+ Commands:queryreplacestring
496K query-replace-string;replace

$497 #498 +499 K500 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.

497$ quick-exit command
498# quick_exit
499+ Commands:quickexit
500K quick-exit;exit;quit

$501 #502 +503 K504 quote-character

Default binding: ^Q

Syntax:

```
n   quote
```

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

If a positive numeric arguments 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.

501$ quote-character command
502# quote_character
503+ Commands:quotecharacter
504K quote-character

$505 #506 +507 K508 read-file

Default binding: ^X^R

Syntax:

```
read-file   file name
```

This command reads the named file into the current buffer, 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 numeric arguments.

505$ read-file command
506# read_file
507+ Commands:readfile
508K read-file;read;file

$509 #510 +511 K512 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 *n*th 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.

509$ redraw-display command
510# redraw_display
511+ Commands:redrawdisplay
512K redraw-display;window

$513 #514 +515 K516 # remove-mark

Default binding: ^X_ (Ctrl+X Spacebar)

Syntax:

```
n   remove-mark
```

This command eliminates the mark number *n*.

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

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

513$ remove-mark command
514# remove_mark
515+ Commands:removemark
516K remove-mark;mark

$517 #518 +519 K520 rename-screen

No default binding.

Syntax:

```
rename-screen   new name
```

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

This command is unaffected by numeric arguments.

517$ rename-screen command
518# rename_screen
519+ Commands:renamescreen
520K rename-screen;screen

$521 #522 +523 K524 replace-string

Default binding: M-R

Syntax:

```
n  replace-string  pattern  replacement
```

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

If no numeric arguments 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 $sterm variable (it is usually the Meta key). Also, both strings may have default values (which are stored in the $search and $replace variables). If you want to replace a string with nothing, and there is a non-empty default for the *replacement* string, striking ^K will override that default and enter an empty string instead.

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

521$ replace-string command
522# replace_string
523+ Commands:replacestring
524K replace-string;replace

$525 #526 +527 K528 resize-window

Default binding: ^XW

Syntax:

```
n   resize-window
```

If *n* is a positive number, this command changes the height of the current window 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 screen, 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 numeric arguments is specified, nothing happens.

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

525$ resize-window command
526# resize_window
527+ Commands:resizewindow
528K resize-window;resize;window

$529 #530 +531 K532 restore-screen

No default binding.

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

This command is unaffected by numeric arguments.

529$ restore-screen command
530# restore_screen
531+ Commands:restorescreen
532K restore-screen;screen

$533 #534 +535 K536 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.

533$ restore-window command
534# restore_window
535+ Commands:restorewindow
536K restore-window;save-window

$_{537}$ #$_{538}$ +$_{539}$ K$_{540}$ **reverse-incremental-search**

Default binding: ^XR

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

While searching towards the beginning of the buffer, each successive character leaves the point at the beginning of the matched string. Typing a ^R causes the next occurrence of the string to be searched for (where the next occurrence does not overlap the current occurrence). A ^S changes the direction to a forward search (as performed by an incremental-search command), pressing the meta key terminates the search and ^G aborts the operation. Pressing the Backspace key (or using ^H) 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. MAGIC 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 numeric arguments.

537$ reverse-incremental-search command
538# reverse_incremental_search
539+ Commands:reverseincrementalsearch
540K reverse-incremental-search;search

$541 #542 +543 K544 # save-file

Default binding: ^X^S

This command writes the contents of the current buffer to disk, if the buffer's contents have been changed since the last read or write operation or the last invocation of the unmark-buffer 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 find-file, read-file, write-file or change-file-name command), the save-file command fails.

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

This command is unaffected by numeric arguments.

541$ save-file command
542# save_file
543+ Commands:savefile
544K save-file;save;file;write

$545 #546 +547 K548 save-window

No default binding.

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

This command is unaffected by numeric arguments.

545$ save-window command
546# save_window
547+ Commands:savewindow
548K save-window;restore-window

$549 #550 +551 K552 scroll-next-down

Default binding: M-^V

Syntax:

```
scroll-next-down
```

or:

```
n   scroll-next-down
```

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

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

549$ scroll-next-down command
550# scroll_next_down
551+ Commands:scrollnextdown
552K scroll-next-down;next-page

$553 #554 +555 K556 scroll-next-up

Default binding:

Syntax:

```
scroll-next-up
```

or:

```
n   scroll-next-up
```

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

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

553$ scroll-next-up command
554# scroll_next_up
555+ Commands:scrollnextup
556K scroll-next-up;previous-page

$557 #558 +559 K560 search-forward

Default binding: ^S

Syntax:

```
n   search-forward   search string
```

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

If a matching text is found in the buffer, the point 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 search-reverse command had been invoked with the corresponding positive number (-*n*).

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

Note: the *search string* becomes the value of the $search variable

557$ search-forward command
558# search_forward
559+ Commands:searchforward
560K search-forward;search

$561 #562 +563 K564 search-reverse

Default binding: ^R

Syntax:

```
n   search-reverse   search string
```

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

If a matching text is found in the buffer, the point 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 search-forward command had been invoked with the corresponding positive number (-*n*).

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

Note: the *search string* becomes the value of the $search variable

561$ search-reverse command
562# search_reverse
563+ Commands:searchreverse
564K search-reverse;search

$565 #566 +567 K568 select-buffer

Default binding: ^XB

Syntax:

```
select-buffer   buffer
```

or:

```
n   select-buffer   buffer
```

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

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

565$ select-buffer command
566# select_buffer
567+ Commands:selectbuffer
568K select-buffer;buffer

$569 #570 +571 K572 # set

Default binding: ^X^A

Syntax:

```
set   variable   value
```

or:

```
n   set   variable
```

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

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

If *value* is the string "ERROR", the command fails (this allows detection of error cases when *value* is actually a function.

569$ set command
570# set
571+ Commands:set
572K set;variable

$573 #574 +575 K576 set-encryption-key

Default binding: M-E

Syntax:

```
set-encryption-key   key
```

This command sets the current buffer's encryption key (used when the buffer is in CRYPT 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 numeric arguments.

573$ set-encryption-key command
574# set_encryption_key
575+ Commands:setencryptionkey
576K set-encryption-key;encryption

$577 #578 +579 K580 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*.

577$ set-fill-column command
578# set_fill_column
579+ Commands:setfillcolumn
580K set-fill-column;fill

$[581] #[582] +[583] K[584] **set-mark**

Default bindings: M-__ (Esc Spacebar) and M-.

Syntax:

```
n   set-mark
```

This command sets the mark number *n* at the point.

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

581$ set-mark command
582# set_mark
583+ Commands:setmark
584K set-mark;mark

$585 #586 +587 K588 shell-command

Default binding: ^X!

Syntax:

```
shell-command   program
```

or:

```
n   shell-command   program
```

This command 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 DOS box. The current working directory where the *program* executes is set to the directory of the file in the current window (or, if that window is not associated to a filename, to the last visited directory).

If no numeric argument is specified, MicroEMACS and the launched *program* run independently. If a numeric argument is specified, MicroEMACS synchronizes with the *program*.

Note: Under MS-Windows 3.x, you cannot use this command to launch a Windows application. Use execute-program instead.

585$ shell-command command
586# shell_command
587+ Commands:shellcommand
588K shell-command;execute;spawn;shell;DOS

$589 #590 +591 K592 show-files

No default binding

Syntax:

```
show-files   starname
```

This command 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 numeric arguments.

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

589$ show-files command
590# show_files
591+ Commands:showfiles
592K show-files;file

$593 #594 +595 K596 shrink-window

Default binding: ^X^Z

Syntax:

```
n   shrink-window
```

If *n* is a positive number, this command decreases the height of the current window by *n* lines. The window located immediately below the current window (or, if the current window is at the bottom of the screen, 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 grow-window command had been invoked with the corresponding positive number (-*n*).

If no numeric arguments 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 resize-window command.

593$ shrink-window command
594# shrink_window
595+ Commands:shrinkwindow
596K shrink-window;resize;window

$597 #598 +599 K600 split-current-window

Default binding: ^X2

Syntax:

```
n   split-current-window
```

This command splits the current window into two windows. Both windows view the current buffer at the current point.

If a numeric arguments 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 point 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 screen of extraneous windows, use the delete-window or the delete-other-windows commands.

597$ split-current-window command
598# split_current_window
599+ Commands:splitcurrentwindow
600K split-current-window;window

$601 #602 +603 K604 store-macro

No default binding

Syntax:

```
n   store-macro
    contents
       of
    macro
!endm
```

This command stores the commands and directives that follow it, up to the next !ENDM directive, into a "numbered macro". That macro can be invoked later by the execute-macro-*n* command.

A numeric arguments must be specified and it must be a number from 1 to 40. Otherwise, the command fails.

601$ store-macro command
602# store_macro
603+ Commands:storemacro
604K store-macro;macro

$605 #606 +607 K608 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.

605$ store-procedure command
606# store_procedure
607+ Commands:storeprocedure
608K store-procedure;macro

$609 #610 +611 K612 tile-screens

No default binding

Syntax:

```
n   tile-screens
```

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

If a numeric arguments 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.

609$ tile-screens command
610# tile_screens
611+ Commands:tilescreens
612K tile-screens;MDI;screen

$[613] #[614] +[615] K[616] transpose-characters

Default binding: ^T

This command swaps the character that is before the point 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 numeric arguments.

613$ transpose-characters command
614# transpose_characters
615+ Commands:transposecharacters
616K transpose-characters

$617 #618 +619 K620 # trim-region

Default binding: ^X^T

Syntax:

```
trim-region
```

or:

```
n  trim-region
```

This command causes all the trailing space and tab characters between the column position of the point and the end of the processed lines to be deleted.

If a numeric arguments 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 region.

617$ trim-region command
618# trim_region
619+ Commands:trimregion
620K trim-region;region

$621 #622 +623 K624 unbind-key

Default binding: M-^K

Syntax:

```
unbind-key   keystroke
```

This command removes the association between a *keystroke* and a macro or a command, thus destroying a binding.

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

This command is unaffected by numeric arguments.

621$ unbind-key command
622# unbind_key
623+ Commands:unbindkey
624K unbind-key;binding

$625 #626 +627 K628 unbind-menu

No default binding

Syntax:

```
unbind-menu   menu name
```

This command is available only under Microsoft Windows. It destroys a menu item. The *menu name* is specified using the menu name syntax.

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 numeric arguments.

625$ unbind-menu command
626# unbind_menu
627+ Commands:unbindmenu
628K unbind-menu;binding;menu

$629 #630 +631 K632 undent-region

Default binding: M-(

Syntax:

```
n   undent-region
```

This command deletes the first *n* tab characters in front of each line within the current region.

If the numeric argument *n* is not specified, the first tab of each line is deleted.

Note: this command is often used to undo the effect of an indent-region command.

629$ undent-region command
630# undent_region
631+ Commands:undentregion
632K undent-region;tabs;region

$633 #634 +635 K636 universal-argument

Default binding: ^U

This is a dummy command meant to be used in combination with the bind-to-key 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 numeric argument, much as if the meta key had been pressed. Also, each further action on the universal argument key multiplies the existing numeric argument by 4.

633$ universal-argument command
634# universal_argument
635+ Commands:universalargument
636K universal-argument;argument

$637 #638 +639 K640 unmark-buffer

Default binding: M-~

This command clears the change flag of the current buffer. 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 append-file, find-file, read-file, save-file, view-file or write-file).

This command is unaffected by numeric arguments.

Note: the change flag of the current buffer can also be accessed via the $cbflags variable.

637$ unmark-buffer command
638# unmark_buffer
639+ Commands:unmarkbuffer
640K unmark-buffer;buffer

$641 #642 +643 K644 update-screen

No default binding

This command immediately updates all elements of the MicroEMACS display during the execution of a macro. It has no visible effect when used interactively.

This command is unaffected by numeric arguments.

641$ update-screen command
642# update_screen
643+ Commands:updatescreen
644K update-screen;display

$645 #646 +647 K648 # view-file

Default binding:

Syntax:

```
find-file   file name
```

If the named file is already loaded somewhere in the editor, this command brings its buffer up in the current window. 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 VIEW mode.

This command is unaffected by numeric arguments.

645$ view-file command
646# view_file
647+ Commands:viewfile
648K view-file;file;open;read;VIEW

$649 #650 +651 K652 widen-from-region

Default binding: ^X>

This command causes all the invisible text in the narrowed buffer becomes accessible and visible again.

This command is unaffected by numeric arguments.

649$ widen-from-region command
650# widen_from_region
651+ Commands:widenfromregion
652K widen-from-region;region;buffer;scope

$_{653}$ #$_{654}$ +$_{655}$ K$_{656}$ **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.

653$ wrap-word command
654# wrap_word
655+ Commands:wrapword
656K wrap-word;word

$_{657}$ #$_{658}$ +$_{659}$ K$_{660}$ write-file

Default binding: ^X^W

Syntax:

```
write-file   file name
```

This command writes the contents of the current buffer to disk, using the specified *file name*. This file name becomes the one associated with the buffer (indicated by the $cfname variable).

This command is unaffected by numeric arguments.

657$ write-file command
658# write_file
659+ Commands:writefile
660K write-file;file;write;save

$661 #662 +663 K664 write-message or print

No default binding

Syntax:

```
print   message
```

or:

```
write-message   message
```

This command causes the specified *message* to appear on the message line.

This command is unaffected by numeric arguments.

661$ write-message and print commands
662# write_message
663+ Commands:writemessage
664K write-message;print;message

$665 #666 +667 K668 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.

665$ yank command
666# yank
667+ Commands:yank
668K yank;kill

$669 #670 +671 K672 yank-pop

Default binding: M-Y

Syntax:

```
n   yank-pop
```

This command cycles the kill ring *n* times (as done by the cycle-ring command) and inserts the contents of the kill buffer at the point. If the previous command was yank or yank-pop, the text inserted by that command is deleted before the new text is inserted.

If no numeric argument 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 $yankflag variable.

669$ yank-pop command
670# yank_pop
671+ Commands:yankpop
672K yank-pop;yank;kill