$1 #2 +3 # MicroEMACS for MS-Windows

Index

update 1.1

1$ Index
2# MainIndex
3+ Index:0000

$4 #5 +6 K7 # Introduction

MicroEMACS is a tool for creating and changing documents, programs, and other text files. It is both relatively easy for the novice to use, but also very powerful in the hands of an expert. MicroEMACS can be extensively customized for the needs of the individual user.

MicroEMACS allows several files to be edited at the same time. The display can be split into different windows and screens, and text may be moved freely from one window on any screen to the next. Depending on the type of file being edited, MicroEMACS can change how it behaves to make editing simple. Editing standard text files, program files and word processing documents are all possible at the same time.

There are extensive capabilities to make word processing and editing easier. These include commands for string searching and replacing, paragraph reformatting and deleting, automatic word wrapping, word move and deletes, easy case controlling, and automatic word counts.

For complex and repetitive editing tasks editing macros can be written. These macros allow the user a great degree of flexibility in determining how MicroEMACS behaves. Also, any and all the commands can be used by any keystroke by changing, or rebinding, what commands various keys invoke.

Special features are also available to perform a diverse set of operations such as file encryption, automatic backup file generation, entabbing and detabbing lines, executing operating system commands and filtering of text through other programs (like SORT to allow sorting text).

4$ Introduction
5# Introduction
6+ Index:0005
7K intro;introduction;beginning;basics

$_{8}$ #$_{9}$ +$_{10}$ K$_{11}$ # History

EMACS was originally a text editor written by Richard Stallman at MIT in the early 1970s for Digital Equipment computers. Various versions, rewrites and clones have made an appearance since.

This version of MicroEMACS is derived from code written by David G. Conroy in 1985. Later modifications were performed by Steve Wilhite and George Jones. In December of 1985 Daniel Lawrence picked up the then current source (version 2.0) and made extensive modifications and additions to it over the course of the next six years.

In November 1990, Pierre Perret produced a port of MicroEMACS 3.10e to the Microsoft Windows 3.0 environment (BETA version 0.6a which never enjoyed a full release). The first public version, 1.0, based on MicroEMACS 3.11c, was released in April 1992.

Update 1.1 includes bug fixes, port to Windows NT, support of scroll bars and drag and drop mechanism. It is the first release to include a complete help file.

8$ History
9# History
10+ Index:9010
11K history;author;credits

$12 #13 +14 K15 Support

Updates and support for the current version are available. Commercial support and usage and resale licences are also available.

For questions regarding the official MicroEMACS editor, contact Daniel Lawrence. For technical questions specific to the port of MicroEMACS to the Microsoft Windows environment, contact Pierre Perret.

The home BBS of MicroEMACS is "The Programmer's Room".

12$ Support
13# Support
14+ Index:9020
15K support;bugs;author;credits;licence;updates

$16 #17 +18 K19 Bulletin Board System

The latest executables, sources and documentations can be obtained from:

The Programmer's Room
Opus 201/10
300/1200/2400 and 9600 (Hayes V series only)
(317) 742-5533 no parity 8 data bits no stop bits

16$ Bulletin Board System
17# TheProgrammersRoom
18+ Index:9500
19K BBS;bulletin board;home;support;author;updates;programmer's room;modem

$[20] #[21] +[22] K[23] The current MicroEMACS author can be contacted by writing to:

USMAIL: **Daniel Lawrence**
617 New York Street
Lafayette, IN 47901

UUCP: pur-ee!mdbs!dan

Internet: mdbs!dan@ee.ecn.purdue.edu

The Programmer's Room BBS:
Dan Lawrence

20$ Daniel Lawrence
21# DanielLawrence
22+ Index:9510
23K Lawrence;author;standard;address

$[24] #[25] +[26] K[27] The author of the port of MicroEMACS to the Microsoft Windows environment can be contacted by writing to:

USMAIL: **Pierre Perret**
4326 W Michigan Ave
Glendale, AZ 85308

Internet: P.Perret@az05.bull.com

CompuServe: 73757,2337

BIX: pierre_perret

The Programmer's Room BBS:
Pierre Perret

24$ Pierre Perret
25# PierrePerret
26+ Index:9510
27K Perret;author;Microsoft Windows;MS Windows;address

$28 #29 +30 K31 Copyright

This help file was authored by Pierre Perret.

28$ Copyright
29# Copyright
30+ Index:9030
31K copyright;licence;author;credits

$32 #33 +34 K35 Keyboard

All the MicroEMACS documentation talks about commands and the keystrokes needed to use them. Each MicroEMACS command has a name, and most of them are bound to a sequence of keys. Some of them are bound to mouse actions. The following commands are useful when looking for a binding:

M-?	apropos	looks up commands
	describe-bindings	lists all the bindings
^X?	describe-key	displays the command bound to a keystroke

In this help file, when a command is mentioned, its default key binding is often shown. Note that these bindings can be modified, in particular by the start-up file.

Keystrokes for commands include one of several prefixes, and a command character. Command keystrokes look like these:

^A	hold down Ctrl, press 'A'
M-A	press the meta key, release it and press 'A'
^XA	hold down Ctrl, press 'X', release, press 'A'
^X^A	hold down Ctrl, press 'X', release, hold Ctrl, press 'A'
A-C	hold down Alt, press 'C'
S-FN1	hold down Shift, press function key F1
FN^1	hold down Ctrl, press function key F1

Under Microsoft Windows, key bindings are displayed in front of menu items, using a CUA type syntax instead of the above-mentioned one. Though this may seem inconsistent, it looks more familiar to inexperienced users and is far less cryptic when it comes to special keys (Ins, Del, Arrows...).

32$ Keyboard
33# Keyboard
34+ Index:2010
35K keyboard;binding

$36 #37 +38 K39 Procedures

$40 #41 +42 K43 # The Basics

MicroEMACS is a very powerful tool to use for editing text files. It has many commands, options and features to let you do just about anything you can imagine with text. But don't let this apparent complexity keep you from using it.... MicroEMACS can also be very simple.

To start editing text, all the keys you really need to know are the arrow keys. These keys let you move around in your file.

MicroEMACS also works by using control keys. Here are a few basic commands:

^P	Move upward
^B	Move backward
^F	Move forward
^N	Move downward
^X^S	Save your file
^X^C	Exit MicroEMACS

A hat sign "^" before a key means to hold the Ctrl key down and press the next character. For example, to exit MicroEMACS, hold down the Ctrl key and strike X and then C.

40$ The Basics
41# TheBasics
42+ Procedures:010
43K basics;introduction

$44 #45 +46 K47 # Getting at Files

The way you edit a file within MicroEMACS is by first reading it into a buffer, altering it and then saving it. Therefore, the most commonly used commands to access files are:

^X^F	find-file	to read a file from disk for editing
^X^S	save-file	to save an edited file to disk

Other read commands are:

^X^I	insert-file	to insert at the point
^X^R	read-file	to replace the whole buffer contents
^X^V	view-file	to read for viewing, preventing any alterations

To save a buffer to another file than the one MicroEMACS would normally access, use:

^X^W	write-file	to overwrite the file's previous contents
^X^A	append-file	to append to the end of the file

44$ Getting at Files
45# GettingAtFiles
46+ Procedures:020
47K file;open;save;read;write

$[48] #[49] +[50] K[51] Searching and Replacing

Commands for searching for and replacing strings come in a number of different flavors. The simplest command is:

^S	search-forward

Following that, you can search for yet another occurrence of the same string by using:

A-S	hunt-forward

You can also search towards the beginning of the file instead of towards the end by using:

^R	search-reverse
A-R	hunt-backward

To replace strings, use:

M-R	replace-string	to replace all occurrences
M-^R	query-replace-string	to get queried for each replacement

MicroEMACS also supports incremental searching:

^XS	incremental-search	towards the beginning
^XR	reverse-incremental-search	towards the end

48$ Searching and Replacing
49# SearchingAndReplacing
50+ Procedures:030
51K search;replace

$_{52}$ #$_{53}$ +$_{54}$ K$_{55}$ # Cutting and Pasting

MicroEMACS allows you to manipulate text by blocks of any size. You can copy or move text within MicroEMACS through the kill buffer.

Under Microsoft Windows, you can also exchange text with other Windows applications via the clipboard, using the cut-region, clip-region and insert-clip commands.

52$ Cutting and Pasting
53# CuttingAndPasting
54+ Procedures:040
55K cut;paste;move;kill;copy;clipboard

$[56] #[57] +[58] K[59] Moving Text

To move text from one place to another:

1. Move to the beginning of the text you want to move.
2. Set the mark there with the set-mark (M-) command.
3. Move the point to the end of the text.
4. Use the kill-region (^W) command to delete the region you just defined. The text will be saved in the kill buffer.
5. Move the point to the place you want the text to appear.
6. Use the yank (^Y) command to copy the text from the kill buffer to the current point.

Repeat steps 5 and 6 to insert more copies of the same text.

56$ Moving Text
57# MovingText
58+ CuttingAndPasting:movingtext
59K move

$60 #61 +62 K63 Copying Text

To copy text from one place to another:

1. Move to the beginning of the text you want to copy.
2. Set the mark there with the set-mark (M-) command.
3. Move the point to the end of the text.
4. Use the copy-region (M-W) command to copy the region to the kill buffer.
5. Move the point to the place you want the text to appear.
6. Use the yank (^Y) command to copy the text from the kill buffer to the current point.

Repeat steps 5 and 6 to insert more copies of the same text.

60$ Copying Text
61# CopyingText
62+ CuttingAndPasting:copyingtext
63K copy

$64 #65 +66 K67 # Using the Mouse

MicroEMACS can use the mouse to perform many basic editing tasks. Unless mouse behavior has been altered by a macro, you can perform the following actions:

- Copying a Region
- Killing a Region
- Moving a Mode Line
- Pasting Text
- Repositioning the Point
- Scrolling Text Inside a Window

64$ Using the Mouse
65# UsingTheMouse
66+ Procedures:050
67K mouse

$68 #69 +70 K71 Repositioning the Point with the Mouse

Move the mouse to where you want the point to be, and click once with the left mouse button and release. The point will move there, making any screen or window active to do so.

68$ Repositioning the Point with the Mouse
69# RepositioningThePoint
70+ UsingTheMouse:repositioningthepoint
71K position;point;mouse

$72 #73 +74 K75 Scrolling Text Inside a Window with the Mouse

Position the mouse on the text to drag, press the left button, move the mouse to where to display the text (horizontally or vertically), and release the mouse button.

If you are using the CUA.CMD page (which is usually the case under Microsoft Windows), the above action is performed by pressing the **right** mouse button instead of the left one.

Note that if you drag diagonally and the $diagflag variable is set to FALSE (the default value), the text will move only in the vertical direction.

72$ Scrolling Text Inside a Window with the Mouse
73# ScrollingTextInsideAWindow
74+ UsingTheMouse:scrollingtextinsideawindow
75K scroll;mouse

$[76] #[77] +[78] K[79] Moving a Mode Line with the Mouse

Position the mouse on a mode line (except the bottom one which cannot be moved), press the left button, move to another position and release the button. The mode line will move, resizing the windows which are above and below.

If you are using the CUA.CMD page (which is usually the case under Microsoft Windows), the above action is performed by pressing the **right** mouse button instead of the left one.

76$ Moving a Mode Line with the Mouse
77# MovingAModeLine
78+ UsingTheMouse:movingamodeline
79K mode line;mouse

$80 #81 +82 K83 Copying a Region with the Mouse

Position the mouse at the beginning of the text to be copied, press the right button, move the mouse to the other end of the text, release the button. This actually makes the selected text the current region and then copies it into the kill buffer.

If you are using the CUA.CMD page (which is usually the case under Microsoft Windows), the above action is performed by pressing the **Shift** key and the **right** mouse button together instead of just the right mouse button.

80$ Copying a Region with the Mouse
81# CopyARegion
82+ UsingTheMouse:copyingaregion
83K copy;region;mouse

$84 #85 +86 K87 Killing a Region with the Mouse

After copying a region, without moving the mouse, click the right mouse button once. The text will be deleted, but it will still be kept in the kill buffer.

If you are using the CUA.CMD page (which is usually the case under Microsoft Windows), the above action is performed by pressing the **Shift** key and the **right** mouse button together instead of just the right mouse button.

84$ Killing a Region with the Mouse
85# KillARegion
86+ UsingTheMouse:killingaregion
87K kill;delete;region;mouse

$88 #89 +90 K91 Pasting Text with the Mouse

Move anywhere away from the place the mouse was last clicked, and click the right button once. The last text placed in the kill buffer will be inserted there.

If you are using the CUA.CMD page (which is usually the case under Microsoft Windows), the above action is performed by pressing the **Shift** key and the **right** mouse button together instead of just the right mouse button.

88$ Pasting Text with the Mouse
89# PastingText
90+ UsingTheMouse:pastingtext
91K paste;mouse

$92 #93 +94 K95 # Using menus

Under Microsoft Windows, MicroEMACS sports an extensive menu system. Menu items can point to a pop-up menu or directly invoke a command or a macro. A few menu items are not linked to any MicroEMACS commands or macro (for instance, the "About..." item in the "Help" menu).

The text of each menu item can contain the following hints:

Items that lead to the apparition of a dialog box are followed by an ellipsis "**...**".

Items that require the user to type additional information in the message line are followed by a colon "**:**".

Items that require a numeric argument are preceded by an equal sign "**=**".

Items that are equivalent to a key binding have the corresponding key sequence displayed on the right side of the menu.

The MicroEMACS menus can be modified by macros to add/remove menus or menu items. The initial menus on the menu bar are:

File
Edit
Search
Execute
Miscellaneous
Screen
Help

92$ Using Menus
93# UsingMenus
94+ Procedures:060
95K menu

$[96] #[97] +[98] File menu

This menu contains the following items:

Open...	invokes the find-file command. If the MDI.CMD page is loaded, this menu item is modified and bound to the open-file macro
View...	invokes the view-file command
Insert...	invokes the insert-file command
Read over...	invokes the read-file command
Rename...	invokes the change-file-name command
Save	invokes the save-file command
Save as...	invokes the write-file command
Append...	invokes the append-file command
Encryption key :	invokes the set-encryption-key command
Buffer	submenu
Window	submenu
Mode...	brings up a dialog box to change the modes of operation for the current buffer.
Global mode...	brings up a dialog box to change the global modes of operation.
Save + exit	invokes the quick-exit command
Exit	invokes the exit-emacs command

96$ File menu
97# FileMenu
98+ UsingMenus:050

$99 #100 +101 Buffer submenu

This menu is accessed via the File menu. It contains the following items:

Next	invokes the next-buffer command
Select :	invokes the select-buffer command
Unmark	invokes the unmark-buffer command
Rename :	invokes the name-buffer command
Delete :	invokes the delete-buffer command
Narrow to region	invokes the narrow-to-region command
Widen from region	invokes the widen-from-region command
List	invokes the list-buffers command

99$ Buffer submenu
100# BufferSubmenu
101+ FileMenu:050

$[102] #[103] +[104] Window submenu

This menu is accessed via the File menu. It contains the following items:

Split	invokes the split-current-window command
Delete	invokes the delete-window command
Delete others	invokes the delete-other-windows command
Next	invokes the next-window command
Previous	invokes the previous-window command
Scroll	submenu
Size	submenu

102$ Window submenu
103# WindowSubmenu
104+ FileMenu:060

$105 #106 +107 Window Scroll submenu

This menu is accessed via the Window submenu of the File menu. It contains the following items:

= Up	invokes the move-window-up command
= Down	invokes the move-window-down command
= Next up	invokes the scroll-next-up command
= Next down	invokes the scroll-next-down command

105$ Window Scroll submenu
106# ScrollSubmenu
107+ WindowSubmenu:050

$108 #109 +110 Window Size submenu

This menu is accessed via the Window submenu of the File menu. It contains the following items:

= Grow	invokes the grow-window command
= Shrink	invokes the shrink-window command
= Height	invokes the resize-window command

108$ Window Size submenu
109# WSizeSubmenu
110+ WindowSubmenu:060

$[111] #[112] +[113] Edit menu

This menu contains the following items:

Clipboard	submenu
Mark	submenu
Yank	invokes the yank command
Region	submenu
Paragraph	submenu
Line	submenu
Word	submenu
Delete blank lines	invokes the delete-blank-lines command
Transpose characters	invokes the transpose-characters command
Tab	invokes the handle-tab command
Quote	invokes the quote-character command
= Fill column	invokes the set-fill-column command. The emacs.rc page modifies this menu item slightly so that it prompts you for the fill column value.

If the CUA.CMD page is loaded, the menu is modified by the addition of the following item (before "Region"):

Selection	submenu

111$ Edit menu
112# EditMenu
113+ UsingMenus:060

$[114] #[115] +[116] Clipboard submenu

This menu is accessed via the Edit menu. It contains the following items:

Cut region	invokes the cut-region command
Copy region	invokes the clip-region command
Paste	invokes the insert-clip command

If the CUA.CMD page is loaded, the menu is modified and, instead, contains the following items:

Cut	deletes and copies to the clipboard the text contained in the current selection
Copy	copies (without deleting) to the clipboard the text contained in the selection
Paste	inserts the text from the clipboard at the point

114$ Clipboard submenu
115# ClipboardSubmenu
116+ EditMenu:010

$[117] #[118] +[119] Mark submenu

This menu is accessed via the Edit menu. It contains the following items:

Set	invokes the set-mark command
Remove	invokes the remove-mark command
Exchange	invokes the exchange-point-and-mark command

117$ Mark submenu
118# MarkSubmenu
119+ EditMenu:020

$120 #121 +122 Selection submenu

This menu is accessed via the Edit menu when the CUA.CMD page is loaded. It contains the following items:

Upper case	converts all the selected text to upper case
Lower case	converts all the selected text to lower case
Count words	displays on the message line the number of words, characters and lines that compose the selected text
Flip	exchanges the point with the other end of the selection
Select region	makes the current region the current selection

120$ Selection submenu
121# SelectionSubmenu
122+ EditMenu:025

$123 #124 +125 Region submenu

This menu is accessed via the Edit menu. It contains the following items:

Kill	invokes the kill-region command
Copy	invokes the copy-region command
Upper case	invokes the case-region-upper command
Lower case	invokes the case-region-lower command
Entab	invokes the entab-region command
Detab	invokes the detab-region command
Trim	invokes the trim-region command
Indent	invokes the indent-region command
Undent	invokes the undent-region command
Count words	invokes the count-words command

123$ Region submenu
124# RegionSubmenu
125+ EditMenu:030

$126 #127 +128 Edit Paragraph submenu

This menu is accessed via the Edit menu. It contains the following items:

Kill	invokes the kill-paragraph command
Fill	invokes the fill-paragraph command

126$ Edit Paragraph submenu
127# EParagraphSubmenu
128+ EditMenu:040

$129 #130 +131 Edit Line submenu

This menu is accessed via the Edit menu. It contains the following items:

Kill to end	invokes the kill-to-end-of-line command
Open	invokes the open-line command

129$ Edit Line submenu
130# ELineSubmenu
131+ EditMenu:050

$132 #133 +134 Edit Word submenu

This menu contains the following items:

Kill next	invokes the delete-next-word command
Kill previous	invokes the delete-previous-word command
Capitalize	invokes the case-word-capitalize command
Lower case	invokes the case-word-lower command
Upper case	invokes the case-word-upper command

132$ Edit Word submenu
133# EWordSubmenu
134+ EditMenu:060

$[135] #[136] +[137] Search menu

This menu contains the following items:

Search forward :	invokes the search-forward command
Search backward :	invokes the search-reverse command
Hunt forward	invokes the hunt-forward command
Hunt backward	invokes the hunt-backward command
Incremental search :	invokes the incremental-search command
Reverse incremental :	invokes the reverse-incremental-search command
Replace :	invokes the replace-string command
Query replace :	invokes the query-replace-string command
Goto	submenu
Page	submenu
Paragraph	submenu
Line	submenu
Word	submenu

135$ Search menu
136# SearchMenu
137+ UsingMenus:070

$[138] #[139] +[140] Goto submenu

This menu is accessed via the Search menu. It contains the following items:

Mark	invokes the goto-mark command
Line	invokes the goto-line command
Matching fence	invokes the goto-matching-fence command
Beginning of file	invokes the beginning-of-file command
End of file	invokes the end-of-file command

138$ Goto submenu
139# GotoSubmenu
140+ SearchMenu:010

$[141] #[142] +[143] Page submenu

This menu is accessed via the Search menu. It contains the following items:

Next	invokes the next-page command
Previous	invokes the previous-page command

141$ Page submenu
142# PageSubmenu
143+ SearchMenu:020

$[144] #[145] +[146] Search Paragraph submenu

This menu is accessed via the Search menu. It contains the following items:

Next	invokes the next-paragraph command
Previous	invokes the previous-paragraph command

144$ Search Paragraph submenu
145# SParagraphSubmenu
146+ SearchMenu:030

$[147] #[148] +[149] Search Line submenu

This menu is accessed via the Search menu. It contains the following items:

Next	invokes the next-line command
Previous	invokes the previous-line command
Beginning of	invokes the beginning-of-line command
End of	invokes the end-of-line command

147$ Search Line submenu
148# SLineSubmenu
149+ SearchMenu:040

$150 #151 +152 Search Word submenu

This menu is accessed via the Search menu. It contains the following items:

Next	invokes the next-word command
Previous	invokes the previous-word command
End of	invokes the end-of-word command

150$ Search Word submenu
151# SWordSubmenu
152+ SearchMenu:050

$[153] #[154] +[155] Execute menu

This menu contains the following items:

Windows program :	invokes the execute-program command
Shell program :	invokes the shell-command command
Pipe-in :	invokes the pipe-command command
Filter :	invokes the filter-buffer command
Shell	invokes the i-shell command
EMACS command	submenu
Keyboard macro	submenu
Abort command	invokes the abort-command command

If the DEV.CMD page is loaded, the menu is modified by the addition of the following item:

Make	invokes the run-makefile macro.

153$ Execute menu
154# ExecuteMenu
155+ UsingMenus:080

$156 #157 +158 EMACS command submenu

This menu is accessed via the Execute menu. It contains the following items:

Named command :	invokes the execute-named-command command
Command line :	invokes the execute-command-line command
Procedure :	invokes the execute-procedure command
Buffer :	invokes the execute-buffer command
File...	invokes the execute-file command

156$ EMACS command submenu
157# EMACSCOMMANDSUBMENU
158+ ExecuteMenu:050

$159 #160 +161 Keyboard macro submenu

This menu is accessed via the Execute menu. It contains the following items:

Play	invokes the execute-macro command
Start recording	invokes the begin-macro command
End recording	invokes the end-macro command

159$ Keyboard macro submenu
160# KeyboardMacroSubmenu
161+ ExecuteMenu:060

$[162] #[163] +[164] Miscellaneous menu

This menu contains the following items:

Key bindings	submenu
Menu bindings	submenu
Variable	submenu
Show position	invokes the buffer-position command

162$ Miscellaneous menu
163# MiscellaneousMenu
164+ UsingMenus:090

$[165] #[166] +[167] Key bindings submenu

This menu is accessed via the Miscellaneous menu. It contains the following items:

Bind to Command	invokes the bind-to-key command
Bind to Macro	invokes the macro-to-key command
Unbind	invokes the unbind-key command
Describe key	invokes the describe-key command
List	invokes the describe-bindings command

165$ Key bindings submenu
166# KeyBindingsSubmenu
167+ MiscellaneousMenu:050

$[168] #[169] +[170] Menu bindings submenu

This menu is accessed via the Miscellaneous menu. It contains the following items:

Bind to Command	invokes the bind-to-menu command
Bind to Macro	invokes the macro-to-menu command
Unbind	invokes the unbind-menu command

168$ Menu bindings submenu
169# MenuBindingsSubmenu
170+ MiscellaneousMenu:060

$[171] #[172] +[173] Variable submenu

This menu is accessed via the Miscellaneous menu. It contains the following items:

Set	invokes the set command
Display	invokes the display command
List	invokes the describe-variables command

171$ Variable submenu
172# VariableSubmenu
173+ MiscellaneousMenu:070

$[174] #[175] +[176] Screen menu

This menu contains the following items:

Cascade	invokes the cascade-screens command
Tile	submenu
Arrange Icons	causes iconized screens to be rearranged at the bottom left of the MicroEMACS frame window.
Open	invokes the find-screen command
Rename	invokes the rename-screen command
Size	submenu
Font...	brings up a dialog box to change the font used by MicroEMACS

If the MDI.CMD page is loaded, the menu is modified by the addition of the following items:

Rebuild	rebuilds the set of screens, to have a screen associated with each editing buffer
Kill	deletes the current screen and release the corresponding buffer.

Additional items are added dynamically at the end of the "Screen" menu, listing the available screens. This allows quick switching between those screens.

174$ Screen menu
175# ScreenMenu
176+ UsingMenus:100

$[177] #[178] +[179] Tile submenu

This menu is accessed via the Screen menu. It contains the following items:

Horizontally	causes all non-iconic screens to be rearranged in a tiling scheme, side by side if possible
Vertically	causes all non-iconic screens to be rearranged in a tiling scheme, on top of each other if possible

177$ Tile submenu
178# TileSubmenu
179+ ScreenMenu:050

$[180] #[181] +[182] Screen Size submenu

This menu is accessed via the Screen menu. It contains the following items:

= Height	invokes the change-screen-size command
= Width	invokes the change-screen-width command
Normalize	causes the current screens to be resized so that it is as small as possible while retaining the same height and width in characters.

If the MDI.CMD page is loaded, the menu is modified by the replacement of "= Height" and "= Width" by the following item:

Set:	prompts you for the width and height of the screen, supplying the current values as defaults.

180$ Screen Size submenu
181# SSizeSubmenu
182+ ScreenMenu:060

$[183] #[184] +[185] Help menu

This menu contains the following items:

Index	brings up this help file, on the main index.
Keyboard	brings up this help file, on the keyboard topic
Commands	brings up this help file, on the commands topic
Procedures	brings up this help file, on the procedures topic
List	submenu
Apropos :	invokes the apropos command
Describe key :	invokes the describe-key command
Display variable :	invokes the display command
About...	brings up a dialog box giving some information about MicroEMACS and the people involved in its making.

If the DEV.CMD page is loaded, the menu is modified by the addition of items (before "List") that invoke the Windows help engine for, respectively, Windows 3.0, Windows 3.1 or Win32 Software Development Kits or for Turbo C++. Each of those attempt to select a help topic based on the word currently at the point. You can eliminate the undesired items among these by editing the `macro-to-menu` commands in the DEV.CMD file.

183$ Help menu
184# HelpMenu
185+ UsingMenus:110

$[186] #[187] +[188] List submenu

This menu is accessed via the Help menu. It contains the following items:

Key bindings	invokes the describe-bindings command
Functions	invokes the describe-functions command
Variables	invokes the describe-variables command
Buffers	invokes the list-buffers command

186$ List submenu
187# ListSubmenu
188+ HelpMenu:050

$189 #190 +191 K192 # Customizing Command Keys

MicroEMACS lets you decide what keys activate what command or macro through the use of:

M-K	bind-to-key
^X^K	macro-to-key
M-^K	unbind-key

These commands can be used to permanently change your key bindings by placing them in your start up file. For example, if you have one of those nasty keyboards with a tilde "~" in the upper left corner, where the Escape key should be, and you want the tilde to become the meta key, add this line to emacs.rc:

```
bind-to-key    meta-prefix    ~
```

You can use this to make MicroEMACS feel similar to another editor by changing what keys activate which commands.

The unbind-key command is useful if you have a function key you keep tripping over, or if you are trying to make MicroEMACS look like a much more minimalist editor.

You can get a list of all the key bindings that MicroEMACS uses by using the describe-bindings command. Just do M-X and type:

```
describe-bindings
```

189$ Customizing Command Keys
190# CustomizingCommandKeys
191+ Procedures:070
192K keyboard;binding;command;macro

$193 #194 +195 K196 # Issuing Commands

Commands within MicroEMACS have descriptive names which you can use to invoke them, or bind them to a keystroke or a menu. To invoke one of these commands by name, you can use:

M-X execute-named-command

You can supply numeric arguments to a such a command by prefixing it. You can also use a command line invocation.

To get a list of all the commands in your current MicroEMACS, do M-X and type:

```
describe-bindings
```

The describe-bindings command will display a paged list of all legal commands and the keystrokes to use to invoke them.

193$ Issuing Commands
194# IssuingCommands
195+ Procedures:080
196K command

$197 #198 K199 Interactive Numeric Arguments

Some commands take a number as an argument. For example, to move to a particular line within a file, you use the goto-line (M-G) command. To go to a particular line, precede the command with a number by striking the meta key, typing a number, and then the keys bound to the command. To go to the 123rd line of a file, use:

```
Meta 123    Meta g
```

If a command does not need a numeric argument, it is usually taken as a repeat count. This also works when typing any character. To make a line of 50 dashes type:

```
Meta 50    -
```

197$ Interactive Numeric Arguments
198# NumericArguments
199K repeat;argument

$200 #201 +202 K203 Command Lines

execute-command-line (M-^X) lets you type in a full command line. MicroEMACS macros are made from sequences of these command lines. A command line has three parts:

```
Numeric argument    Command    Arguments
```

The numeric argument is optional and has the same effect as an interactive numeric argument prefixing an interactive invocation of the same command.

Arguments following the command are not always required. If needed arguments have been omitted, the user will be prompted for them on the message line.

To insert the string "<*><*><*>" at the point, do M-^X and then:

```
3    insert-string    "<*>"
```

or to set the current fill column to 64, do M-^X and then:

```
64    set-fill-column
```

200$ Command Lines
201# CommandLines
202+ :commandlines
203K command

$204 #205 +206 K207 # The Outside World

The following commands let you interact with the Operating System or with other applications:

^X^C	exit-emacs	terminates MicroEMACS
M-Z	quick-exit	same as above, but saves all changed buffers first
^X!	shell-command	executes a program within an Operating System "shell"
^X$	execute-program	launches another application
^X@	pipe-command	pipes a program's output into a buffer
^X#	filter-buffer	filters a buffer through a program
^XC	i-shell	opens an Operating System "shell"

204$ The Outside World
205# TheOutsideWorld
206+ Procedures:090
207K execute;spawn;filter;pipe;shell;DOS

$208 #209 K210 Synchronizing With Another Program

When the pipe-command or the filter-buffer commands are used under Microsoft Windows, MicroEMACS creates a DOS box (or "shell box" under Windows NT) and waits for it to terminate.Also, if the execute-program or the shell-command command is invoked with a numeric argument, MicroEMACS waits for the launched application to terminate.

You can cancel the wait by pressing the Esc key or clicking on the "Cancel" button. Note that doing so does not terminate the other program.

For synchronization to work with a DOS box, the DOSExec profile must be set properly. Under Windows NT, shell boxes can be parametrized by setting the Shell and the ShellExecOption profiles.

208$ Synchronizing With Another Program
209# SynchronizingWithADosBox
210K DOS;pipe;filter;synchronize

$211 #212 +213 K214 # Buffers

A buffer is where MicroEMACS stores text. Normally that text is read from a file, and is visible in an editing window. But text stored in buffers can also be MicroEMACS macros, temporary storage for macros, or lists of screens, files, buffers, variables, commands or bindings created by MicroEMACS commands. Commands that deal with buffers include:

^XB	select-buffer
^XK	delete-buffer
^X^B	list-buffers
^XX	next-buffer

211$ Buffers
212# Buffers
213+ Procedures:100
214K buffer

$215 #216 +217 K218 # Regions

Regions are used in MicroEMACS to specify what text is acted on by many commands. A region is defined as all the text between the point, and the last placed mark. To define a region:

1. Move the point to the beginning of the text you want to effect
2. Use the set-mark (M-) command to position the mark at the current point
3. Move the point to the end of the text you want to affect

At this time, the text between the mark and the point is the current region which will be affected by many commands. Regions can be defined backwards as well as forwards, and can include the entire buffer, or as little as one character.

215$ Regions
216# Regions
217+ Procedures:110
218K region

$219 #220 +221 K222 # Paragraphs

MicroEMACS defines a paragraph as any group of lines of text surrounded by blank lines. A line starting with one of the characters in the $paralead variable is considered the first line of a paragraph. Also, if line starts with one of the characters in the $fmtlead variable, the following line is considered to be the beginning of a paragraph.

Commands that deal with paragraphs include:

M-N	next-paragraph
M-P	previous-paragraph
M-^W	kill-paragraph
M-Q	fill-paragraph

219$ Paragraphs
220# Paragraphs
221+ Procedures:120
222K paragraph

$223 #224 +225 K226 # Words

Words are defined, by default, as a string of characters consisting of alphabetic, numeral and the underscore "_" character. You can change this by setting the $wchars variable to a list of all the characters you want considered as part of a word.

The commands that deal with words include:

M-F	next-word
M-B	previous-word
M-D	delete-next-word
M-^H	delete-previous-word
M-^C	count-words

223$ Words
224# Words
225+ Procedures:130
226K word

$227 #228 +229 K230 # Screens

A screen is a collection of windows which are displayed together. On some non-graphically oriented systems, only one screen is displayed at a time. Under other graphical oriented operating systems like Microsoft Windows, X-Windows, the Macintosh or the Amiga, each screen may be displayed in an operating system "window". Notice that the MicroEMACS usage of the word window is different from the meaning used in these graphical systems:

MicroEMACS	Operating System
Window	Pane
Screen	Window

Each screen has its own set of windows. Switching from one screen to another (for instance by clicking on that screen) will preserve the window setup, the colors and the buffers being displayed.

When MicroEMACS starts up, it displays a single screen named "MAIN". Extra screens can be created by the command:

A-F find-screen

227$ Screens
228# Screens
229+ Procedures:140
230K screen

$231 #232 +233 K234 # Windows

MicroEMACS uses windows to display and allow you to edit the contents of buffers. A single screen will show one or more windows, separated by a mode line which describes the contents of the window above it.

You can scroll text vertically and horizontally within a window by using the arrow keys or the page-up, page-down, home and end keys. Note that if a line of text extends beyond the boundary of a window, a dollar "**$**" sign is displayed instead of the last visible character.

Here are some window-related commands:

^X2	split-current-window
^X1	delete-other-windows
^X0	delete-window
^XO	next-window
^XP	previous-window

Notice that the MicroEMACS usage of the word window is different from the meaning used in graphical systems:

MicroEMACS	Operating System
Window	Pane
Screen	Window

231$ Windows
232# Windows
233+ Procedures:150
234K window

$235 #236 +237 K238 # Setting Colors

On systems which are capable of displaying colors, the mode commands can be used to set the background and foreground character colors. Using add-mode (^XM) or delete-mode (^X^M) and typing a lowercase color will set the background color in the current window. An uppercase color will set the foreground color in the current window.

In a similar manner, add-global-mode (M-M) and delete-global-mode (M-^M) will set the background or foreground colors of future windows.

Colors that MicroEMACS knows about are: **white**, **gray** (dark grey), **grey** (light grey), **cyan**, **lcyan** (light cyan), **magenta**, **lmagenta** (light magenta), **yellow**, **lyellow** (light yellow), **blue**, **lblue** (light blue), **red**, **lred** (light red), **green**, **lgreen** (light green) and **black**. If the computer you are running on does not have enough colors, MicroEMACS will attempt to guess at what color to use when you ask for one which is not there (systems with only 8 colors support: white, cyan, magenta, yellow, blue, red, green and black).

Under Microsoft Windows, the whole 16 colors above are available if the display system supports them (depending on the value of the Colors profile). In that case, Mode lines are displayed as black characters on a light grey background. The message line and desktop colors can be modified through the Windows "control panel" as "window text", "window background" and "application workspace". The value of the $deskcolor variable is always irrelevant.

$239 #240 +241 K242 # Setting the Font

Under Microsoft Windows, the font used by MicroEMACS to display text within the screens and the message line can be selected by using the **Font...** item in the **Screen** menu. This brings up a dialog box in which you can select:

The character set	"ANSI" is the usual default within Windows application. "OEM" is useful when displaying files that contain pseudo-graphics characters.
The face name	You can chose any of the available fixed-pitch faces.
The size of the font	You can either chose one of the font heights listed or type one if you have scalable fonts. All heights are expressed in pixels.
The font weight	Normal unless you check the "Bold" box.

A sample of the selected font is shown, specifying its height and width. The maximum screen size is calculated as the number of columns and rows (including mode lines) that would be displayed in a maximized screen when the MicroEMACS frame is maximized.

Pressing the Enter key or the **OK** button effects the change of font in MicroEMACS. Pressing the Alt+S keys or the **Save** button has the same effect, but also saves the font selection in the profiles so that next time MicroEMACS is started, it uses that font. Pressing the Escape key or the **Cancel** button returns to MicroEMACS without changing the font.

239$ Setting the Font
240# Fonts
241+ Procedures:170
242K font

$243 #244 +245 K246 # Case Control

The following commands let you change the case of the word at or following the point:

M-C	case-word-capitalize
M-L	case-word-lower
M-U	case-word-upper

Setting a mark, moving to the other end of the region and using one of these commands will change the case of all the words in the selected region:

^X^L	case-region-lower
^X^U	case-region-upper

243$ Case Control
244# CaseControl
245+ Procedures:180
246K case;uppercase;lowercase;capitals

$_{247}$ #$_{248}$ +$_{249}$ K$_{250}$ # Controlling Tabs

By default, MicroEMACS sets the default tab stops every eighth column. This behavior can be changed (usually within the start-up file).

The behavior of the handle-tab (^I or Tab key) command depends on the numeric argument that is supplied to it:

> With no argument, **handle-tab** inserts space characters or a single tab character to get to the next tab stop, depending on its configuration...
>
> 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 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.

Tab characters can be globally replaced by the appropriate number of spaces by the detab-region (^X^D) command. The reverse, entab-region (^X^E) changes multiple spaces to tab characters.

$251 #252 +253 K254 Repetitive Tasks

To perform any repetitive task, where you have a list of things that need to be changed, for instance one per line, follow these steps:

1) Position the point to the beginning of the line to change
2) Invoke begin-macro (^X()to start recording
3) make the change, staying on that line
4) move to the beginning of the next line
5) Invoke end-macro (^X)) to stop recording

Do execute-macro (^XE) once to test your change on the next line. If it is satisfactory, count how many lines need to yet be changed, strike the meta key followed by that number and ^XE. This causes your change to be made on all the lines.

251$ Repetitive Tasks
252# RepetitiveTasks
253+ Procedures:200
254K repeat;macro;keyboard

$255 #256 +257 K258 # Narrowing Your Scope

Many times you will want to do something to a part of the text when the command works on all the text. Also it is helpful to see or edit just a portion of the text.

This kind of editing can be performed by narrowing the buffer and later restoring the invisible portions, using the following commands:

^X< narrow-to-region
^X> widen-from-region

255$ Narrowing Your Scope
256# NarrowingYourScope
257+ Procedures:210
258K scope;buffer;region

$259 #260 +261 K262 # Creating New Commands

MicroEMACS lets you create your own macros to perform any editing tasks, simple or complex. These macros are written in the MicroEMACS macro language. Macros can be invoked by other macros and they can be bound to keystrokes by the macro-to-key (^X^K) command.

For examples of macros, look at the .CMD files supplied with MicroEMACS for Windows. In that package, EMACS.RC is the file which is executed automatically whenever MicroEMACS is started. and all the ???.CMD files contain the code for each page.

259$ Creating New Commands
260# CreatingNewCommands
261+ Procedures:220
262K command;macro

$263 #264 +265 K266 Customizing Menus

MicroEMACS menus can be modified by the following commands (usually employed in the start-up file):

bind-to-menu	creates a menu item bound to a command
macro-to-menu	creates a menu item bound to a macro
unbind-menu	deletes a menu item

With these three commands, menus are specified by using the MicroEMACS menu name syntax.

263$ Customizing Menus
264# CustomizingMenus
265+ Procedures:230
266K menu;binding;command;macro

$267 #268 +269 K270 Menu Name Syntax

Menu names used by the bind-to-menu, macro-to-menu and unbind-menu commands follow a common syntax. A menu name is composed of menu item names separated by right brackets:

```
>item1>item2>item3
```

When a menu name begins by a right bracket "**>**", it means that the menu item immediately following this right bracket is located within the menu bar. A menu name can also be specified as:

```
item1>item2
```

In this case *item1* is located within the last accessed menu. One or more left brackets "**<**" can appear before the first item, meaning it is located as many levels up in the menu hierarchy:

```
<<item1>item2
```

Notes: The tilde character "~" cannot be used to escape the meaning of the brackets ("**<**" or "**>**") and ampersand "**&**" characters within menu names. The brackets simply cannot be escaped. The ampersand can be escaped (i.e. considered as a real ampersand instead of indicating the underscoring of a character) by using two consecutive ampersands: "**&&**".

It is good practice to enclose menu names in double quotes. This is necessary when there are embedded spaces within a name. Also, when a menu name begins by an ampersand, MicroEMACS may misinterpret it as a function name.

See the examples for a more practical explanation...

267$ Menu Name Syntax
268# MenuName
269+ CustomizingMenus:010menuname
270K menu

$271 #272 +273 K274 Menu Item Syntax

Menu item names are used as parts of menu names. They specify a single menu item within a given popup menu or within the menu bar. A menu item name can be formed of an *item text* and/or an *item index*:

```
item text@item index
```

or:

```
item text
```

or:

```
@item index
```

The *item text* specifies the text of the item that appears within the menu, using an ampersand "**&**" as a prefix for the underlined character. Note that the key binding description, if any, is automatically generated by MicroEMACS and should not be part of the *item text*.

The *item index* is a decimal number that specifies the index of the item within the menu. Indexes start at zero.

If the specified item is being created:

The *item text* is mandatory.

Separators (horizontal lines between parts of a popup menu) are specified by the *item text* being a single dash "**-**". Note that either bind-to-menu or macro-to-menu can be used for this, since the bound command or macro is irrelevant (although it has to be a valid one).

The *item index* can be used to specify the position where the new item will be placed

If the *item index* is not specified, the new item is placed at the end of the menu or just after the item that was used in a previous menu binding command.

If the specified item already exists:

If the item is not a separator, only one of *item text* or *item index* is needed (but both can be specified).

If the item is a separator, the *item index* should be specified but **not** the *item text*.

See the examples for a more practical explanation...

271$ Menu Item Syntax
272# MenuItem
273+ CustomizingMenus:020menuitem
274K menu

$[275] #[276] +[277] K[278] Menu Examples

```
bind-to-menu   forward-character   ">&Search>&Character@15>&Next"
bind-to-menu   nop                     "-"
bind-to-menu   backward-character   "&Previous"
```

This creates a new popup menu named "Character" under the "Search" menu, containing the two items "Next" and "Previous", with a separator (for the sake of the demonstration) between the two.

```
unbind-menu     ">&Search>&Character>@1"
```

removes the above-created separator.

```
macro-to-menu   load-c-page     ">Code &page@4>&Load>&C"
macro-to-menu   load-cpp-page   "C&++"
macro-to-menu   load-p-page     "&Pascal"
macro-to-menu   remove-c-page   "<&Remove>&C"
macro-to-menu   remove-cpp-page    "C&++"
macro-to-menu   remove-p-page   "&Pascal"
bind-to-menu    nop                "<-"
macro-to-menu   remove-all-pages   "Remove &all"
```

This (assuming the specified macros actually exist) creates a new menu "Code page", located between the "Execute" and the "Miscellaneous" menus in the menu bar. This new menu contains the "Load", "Remove" and "Remove all" items, the later being preceded by a separator. Both the "Load" and "Remove" items actually lead to sub-menus that both contain "C", "C++" and "Pascal".

275[$] Menu Examples
276[#] MenuExamples
277[+] CustomizingMenus:030menuexamples
278[K] menu

$279 #280 +281 K282 # Drag and Drop

Under MS-Windows 3.1 and above, MicroEMACS supports a "drag and drop" file-selection mechanism. If you select one or more files in the Windows File Manager and drag them with the mouse, dropping them over MicroEMACS generates a pseudo mouse action: MS! that can be used by binding it to a macro.

For instance, the following command causes a macro named "drop-files" to be invoked every time a group of files is dropped on MicroEMACS:

```
macro-to-key   drop-files   MS!
```

The macro that handles the drag and drop mechanism acquires the necessary information from a buffer named "**Dropped files**":

> The first line of that buffer contains the name of the screen on which the drop occurred. It is empty if the files were not dropped on any specific screen (for instance if they were dropped on the message line).
>
> The second and following lines contain the list of dropped files, one pathname per line.

In addition, the $xpos and $ypos variables are set to the text coordinates where the drop occurred (or to the value 255 if the files were not dropped on any specific screen).

The MDI.CMD page contains a sample macro that handles drag and drop.

279$ Drag and Drop
280# DragAndDrop
281+ Procedures:240
282K drag and drop;file;drop

$283 #284 +285 K286 Modes of Operation

Modes determine how MicroEMACS will treat text. Modes affect the contents of a buffer. Global modes determine the modes of newly created buffers.

^XM	add-mode	Adds a mode to the current buffer
^X^M	delete-mode	Removes a mode from the current buffer
M-M	add-global-mode	Adds a global mode
M-^M	delete-global-mode	Removes a global mode

MicroEMACS's modes are:

ASAVE	Automatically Save
CMODE	Editing C programs
CRYPT	Encryption
EXACT	Character Case during Searches
MAGIC	Regular Expression Pattern Matching
OVER	Overstrike Mode
REP	Replace Mode
VIEW	No Changes Permitted
WRAP	Wrap entered text

283$ Modes of Operation
284# ModesOfOperation
285+ Index:2030
286K mode

$287 #288 +289 K290 ASAVE Mode

When this mode is on, MicroEMACS automatically saves the contents of your current buffer to disk every time you have typed 256 characters. The buffer is saved to the file named on the mode line of the buffer. This mode assures you that you will loose very little text should your computer crash while you are editing. Be sure you are willing to have your original file replaced automatically before you add this mode.

The frequency of saving can be altered by changing the contents of the $asave variable.Use the set (^XA) command like this:

^XA $asave 2048

to tell MicroEMACS to automatically save the current buffer after 2048 characters are typed.

Note: the $acount variable contains the count down to the next auto-save.

287$ ASAVE Mode
288# ASAVE
289+ ModesOfOperation:asave
290K ASAVE;mode;autosave;save

$291 #292 +293 K294 CMODE Mode

This mode is specifically for editing programs written in the C language. When CMODE is active, MicroEMACS will try to anticipate what indentation is needed when the newline (^M or Enter key) command is used. It will always bring a pound sign "#" with only leading white space back to the left margin. It will also attempt to flash the cursor over the proper opening fence character matching any closing fence character (one of "**)}]**") that is typed (the duration of this flashing can be controlled by setting the $tpause variable).

Note that the standard start-up files for MicroEMACS install a macro which checks any file being read into MicroEMACS and sets CMODE if the file ends with a .c or .h extension.

Related command:

M-^F goto-matching-fence

291$ CMODE Mode
292# CMODE
293+ ModesOfOperation:cmode
294K CMODE;mode;C language;brace;parenthesis;bracket

$295 #296 +297 K298 CRYPT Mode

For files of a sensitive nature, MicroEMACS can encrypt text as it is written or read. The encryption algorithm is a Beaufort Cipher with a variant key. This is reasonably difficult to decrypt.

When you write out text, if CRYPT mode is active and there is no encryption key, MicroEMACS will ask:

    Encryption String:

Type in a word or phrase of at least five and up to 128 characters for the encryption to use. If you look at the file which is then written out, all the printable characters have been scrambled.To read such a file later, you can use the **-k** switch when calling up MicroEMACS:

    emacs -k *filename*

and you will be asked the encryption key before the file is read.

You can modify the encryption key by using the set-encryption-key (M-E) command.

Note: previous versions of MicroEMACS used a defective encryption method. For compatibility, you can chose to use the older algorithm by setting the $oldcrypt variable to TRUE.

295$ CRYPT Mode
296# CRYPT
297+ ModesOfOperation:crypt
298K CRYPT;mode;encryption;secret

$299 #300 +301 K302 EXACT Mode

Normally, when using search or replace commands, MicroEMACS ignores the case of letters for comparisons. With EXACT mode set, the case of the characters must be the same for a match to occur.

299$ EXACT Mode
300# EXACT
301+ ModesOfOperation:exact
302K EXACT;mode;search;replace;case

$303 #304 +305 K306 # MAGIC Mode

Normally, MicroEMACS uses the string you type in response to a search or replace command as the string to find. When magic mode is enabled, MicroEMACS considers the string you type as a pattern or template to use in finding a string to match. Many characters in this template have special meaning:

.	any single character, except newline.
[*set***]**	any single character from the bracketed *set*.
^	beginning of a line.
$	end of a line.
\	the next character has no special meaning, take the next character literally (unless it is a parenthesis)
?	the preceding character (or "**.**" or **[***set***]**) is optional.
*	the preceding character (or "**.**" or **[***set***]**) matches zero to many times.
+	the preceding character (or "**.**" or **[***set***]**) matches one to many times.
\(*group***\)**	define a group for the replacement string, or for the &group function.

Some characters in the replacement string can have special meanings:

&	insert all of the text matched by the search.
\	the next character has no special meaning (but see groups below...)
\1 to **\9**	insert the text defined by the n^{th} group in the search string.

303$ MAGIC Mode
304# MAGIC
305+ ModesOfOperation:magic
306K MAGIC;mode;search;replace;regular expressions

$307 #308 +309 K310 OVER Mode

MicroEMACS is normally in what many other editors consider "insert" mode. This means when you strike a character, MicroEMACS makes room for that character in the current line, inserting it between the existing characters. In OVER mode, MicroEMACS instead overwrites characters, replacing the existing character under the point with the character you type. OVER mode will maintain the position of text lined up using tabs while replacing existing text.

Be wary of editing Japanese KANJI characters while in this mode: it is possible to overwrite the first byte of the character, leaving the second byte meaningless and alone. REP mode is more appropriate for such files.

307$ OVER Mode
308# OVER
309+ ModesOfOperation:over
310K OVER;mode;insert;REP

$311 #312 +313 K314 # WRAP Mode

This mode causes the point and the previous word to jump down to the next line when you type a space and are beyond the current fill column. This is normally set to column 72, allowing you to enter text non-stop on a standard screen without bothering to use the return key.

To change the column that text is wrapped past, use the set (^XA) command to change the value of the $fillcol variable, like this:

^XA $fillcol *new_value*

MicroEMACS will then be set to wrap words past column *new_value*.

The $wraphook variable contains the command or macro used to perform word wrapping. By default, it is the wrap-word command.

311$ WRAP Mode
312# WRAP
313+ ModesOfOperation:wrap
314K WRAP;mode;fill;word

$315 #316 +317 K318 VIEW Mode

When in VIEW mode, no command which would change the text is allowed. If you attempt any such command, or try to type in any text, MicroEMACS responds with:

[Key Illegal in View Mode]

This mode is very useful when you want to just look at some existing text, as it will prevent you from changing that text. Also MicroEMACS will not attempt a file lock if a file is read in VIEW mode, allowing you to view files which you don't have write access to, or other people have locked. To launch MicroEMACS and read a file in VIEW mode, use the -v switch:

emacs -v *filename*

315$ VIEW Mode
316# VIEW
317+ ModesOfOperation:view
318K VIEW;mode

$319 #320 +321 K322 # REP Mode

MicroEMACS is normally in what many other editors consider "insert" mode. This means when you strike a character, MicroEMACS makes room for that character in the current line, inserting it between the existing characters. In REP mode, MicroEMACS instead replaces the existing character under the point with the character you type. REP mode will not maintain the position of text which takes up multiple columns using tabs since it will replace a single tab character with the typed character which will not take up the same space on screen. For this purpose, the OVER mode is more appropriate

However, Japanese KANJI characters will correctly replace and be replaced in this mode as the two bytes will be considered together when either style character is used.

319$ REP Mode
320# REP
321+ ModesOfOperation:rep
322K REP;mode;insert;OVER

$323 #324 +325 K326 Start-up

There are different things that can be specified on the MicroEMACS command line to control the way the editor operates. These can be switches, which are a dash "**-**" followed by a letter, and possible other parameters, or a start-up file specifier, which is an at sign "**@**" followed by a file name that overrides the default "EMACS.RC".

Under Microsoft Windows, MicroEMACS also uses some profiles from the WIN.INI file.

323$ Start-up
324# Startup
325+ Index:2050
326K start-up

$327 #328 +329 K330 # Start-up File

When MicroEMACS starts executing, it looks for a start-up file which it will execute as a macro before it reads in any other file. This start-up macro usually redefines some bindings (for instance to use function keys) and loads various useful macros.

The name of the start-up file can be specified on the MicroEMACS command line. By default, it is: EMACS.RC.

Unless the pathname of the start-up file is fully qualified, MicroEMACS searches for the file along the path.

327$ Start-up File
328# StartupFile
329+ Index:2050startupfile
330K start-up;emacs.rc

$331 #332 +333 K334 Command Line Switches

The command line used to launch MicroEMACS looks like this:

```
EMACS.EXE    switches     files to edit
```

The following *switches* can be specified:

@*file*	This causes the named *file* to be executed instead of the standard EMACS.RC file before MicroEMACS reads in any other files. More than one of these can be placed on the command line, and they will be executed in the order that they appear.
-C	The following source files on the command line can be changed (as opposed to being in VIEW mode). This is mainly used to cancel the effects of the -v switch used previously in the same command line.
-E	This flag causes emacs to automatically run the start-up file "error.cmd" instead of emacs.rc. This can be used by compilers for error processing.
-G*num*	Upon entering MicroEMACS, position the cursor at the *num* line of the first file.
-I*var value*	Initialize a MicroEMACS variable with *value*.
-K*key*	This tells MicroEMACS to place the source files in CRYPT mode and read it in using *key* as the encryption key. If no key is listed after the **-K** switch, you will be prompted for a key, and it will not be echoed as it is typed.
-R	This places MicroEMACS in "restricted mode" where any commands allowing the user to read or write any files other than the ones listed on the command line are disabled. Also all commands allowing the user access to the operating system are disabled. This makes MicroEMACS a "safe" environment for use within other applications and especially used as a remote editor for an electronic Bulletin Board System (BBS).
-S*string*	After MicroEMACS is started, it automatically searches for *string* in the first source file.
-V	This tells MicroEMACS that all the following files on the command line should be in VIEW mode to prevent any changes being made to them.

331$ Command Line Switches
332# CommandLineSwitches
333+ Index:2050startupcommandlineswitches
334K switches;start-up

$335 #336 +337 K338 # Profiles

Profiles are entries in the WIN.INI file and are used only under Microsoft Windows. MicroEMACS uses a few profiles, all placed under the "[MicroEMACS]" section, to define the initial window size, the initial font and the path names of some files.

The following profiles can be modified by editing the WIN.INI file:

Colors	number of colors supported by the display.
DOSExec	path name of a PIF file for pipe-command, filter-buffer and i-shell
DOSBox	path name of a PIF file for shell-command
HelpFile	path name of this help file
InitialSize	keywords: "maximize", "minimize" or "optimize"
Shell	path name of the shell executable under Windows NT.
ShellExecOption	command execution option for the shell under Windows NT.
TimeSlice	number of milliseconds of processing before yielding to other applications

The font-related profiles (**FontName**, **FontWeight**, **FontWidth**, **FontHeight** and **CharSet**) are updated by MicroEMACS itself when a font selection is saved.

335$ Profiles
336# Profiles
337+ Index:2050startupprofiles
338K profile;start-up;WIN.INI

$339 #340 +341 K342 Colors Profile

The Colors profile is used to force MicroEMACS to run in either color or monochrome mode. In color mode, the mode lines display back text over a light grey background and editable text is displayed as white on black (these colors can be customized). In monochrome mode, MicroEMACS uses the colors specified by the system (configurable through the Windows Control Panel), using highlighted text for the mode lines.

The value associated to the colors profile is the number of colors supported by the system, or zero (to allow MicroEMACS to automatically determine the proper value). Monochrome mode is assumed for values 1 and 2. Values greater than 2 put MicroEMACS in color mode.

If this profile does not appear in the [MicroEMACS] section of the WIN.INI file, the default value is 0.

Setting this profile is particularly useful on monochrome displays that allow multiple shades of gray (in particular, laptop screens), as MicroEMACS mistakenly believes these to be actual color displays.

339$ Colors Profile
340# ColorsProfile
341+ Profiles:ColorsProfile
342K profile;color

$343 #344 +345 K346 DOSExec Profile

The DOSExec profile specifies the path name of a PIF file used by the pipe-command, filter-buffer and i-shell commands under MS Windows 3.x. This profile is also used when the shell-command command is invoked with a numeric argument.

If this profile does not appear in the [MicroEMACS] section of the WIN.INI file, the file "DOSEXEC.PIF" is searched along the path. This is appropriate if, for instance, that file is located in the directory where the MicroEMACS executable resides.

343$ DOSExec Profile
344# DosExecProfile
345+ Profiles:dosexecprofile
346K profile;DOS;PIF;filter;pipe;shell;WIN.INI

$347 #348 +349 K350 DOSBox Profile

The DOSBox profile specifies the path name of a PIF file used when the shell-command is invoked without a numeric argument under MS Windows 3.x.

If this profile does not appear in the [MicroEMACS] section of the WIN.INI file, the file "DOSBOX.PIF" is searched along the path. This is appropriate if, for instance, that file is located in the directory where the MicroEMACS executable resides.

347$ DOSBox Profile
348# DosBoxProfile
349+ Profiles:dosboxprofile
350K profile;DOS;PIF;shell;WIN.INI

$351 #352 +353 K354 # HelpFile Profile

The HelpFile profile specifies the path name of the Help file for MicroEMACS. It allows proper function of the menu items that call-up this Help file.

The default value is the file "MEWIN.HLP" within the directory where the MicroEMACS executable resides.

351$ HelpFile Profile
352# HelpFileProfile
353+ Profiles:helpfileprofile
354K profile;help;WIN.INI

$355 #356 +357 K358 InitialSize Profile

The InitialSize profile specifies options for the sizing of the initial MicroEMACS frame window. It can be one of the following keywords:

maximize	the frame window fills the whole display
icon or **minimize**	MicroEMACS starts as an icon
optimize	the frame window fills the whole display, except a single row of icons at the bottom.

If the InitialSize profile is not used, the initial size of the MicroEMACS frame window is decided by the operating system.

355$ InitialSize Profile
356# InitialSizeProfile
357+ Profiles:initialsizeprofile
358K profile;size;WIN.INI

$359 #360 +361 K362 Shell and ShellExecOption Profiles

The **Shell** profile specifies the path name of the shell executable used by the pipe-command, filter-buffer, i-shell and shell-command commands under Windows NT. If this profile does not appear in the [MicroEMACS] section of the WIN.INI file, the default path name is "CMD.EXE". This is appropriate if that file is located in a directory that appears in the system path.

The **ShellExecOption** profile specifies the string to be inserted between the string specified by the Shell profile and the actual command to be executed (for pipe-command, filter-buffer and shell-command). If this profile does not appear in the [MicroEMACS] section of the WIN.INI file, the default is " /c ". This is appropriate for "CMD.EXE".

359$ Shell and ShellExecOption Profiles
360# ShellProfile
361+ Profiles:ShellProfile
362K profile;Shell

$363 #364 +365 K366 TimeSlice Profile

Under Microsoft Windows 3.x, when MicroEMACS performs a long operation (reading or writing a large file, searching text, moving large chunks of text to/from the kill buffer or clipboard, killing a buffer, etc...), it allows other applications to run concurrently with itself.

The TimeSlice profile specifies how often MicroEMACS should relinquish the processor: when a long operation is in process, MicroEMACS does not yield to other applications until the number of milliseconds thus specified has elapsed.

The default value is 100 milliseconds.

Notes: Under Windows NT, the preemptive multitasking nature of the operating system alleviates the need for MicroEMACS to voluntarily yield to other applications. The TimeSlice profile is still used to determine how often input (like a command to exit the editor) is checked.

If the animated grinder (replacing the hourglass mouse cursor) is enabled, the TimeSlice profile also determines the time interval between each change of the cursor image.

363$ TimeSlice Profile
364# TimeSliceProfile
365+ Profiles:timesliceprofile
366K profile;timeslice;WIN.INI

$_{367}$ #$_{368}$ +$_{369}$ K$_{370}$ # Memory Usage

The only limit to the number of buffers is the memory of your computer. All the buffers, text, screens and windows use memory for storage.

Under Microsoft Windows, the accessible storage can be rather large, depending on the amount of extended memory installed on you system. If you are running in Windows 3.x 386-enhanced mode, MicroEMACS is able to use virtual memory, allowing you to edit very large files.

Under MSDOS, the AMIGA, the Atari ST, the HP150 and other microcomputers you can estimate the memory used by adding up the size of all the files you want to edit simultaneously, multiply by 1.4, and add 170K for the size of MicroEMACS. This results in the amount of free memory needed to edit these files. Under a MSDOS machine with 574K conventional memory available, you can edit files totaling about 288K in size.

On UNIX, Windows NT and other systems with large virtual memory there is almost no limit to the number and size of files you edit.

367$ Memory Usage
368# MemoryUsage
369+ Index:2060
370K memory

$371 #372 +373 K374 MS-Windows Specifics

The port of MicroEMACS to the Microsoft Windows environment exhibits a few particularities not encountered with other versions of the editor:

All the standard commands are available. Additional commands are available: they allow access to the clipboard, menu customization, invocation of the help engine and control of screens as MDI (Multiple Document Interface) windows.

In interactive mode, the file access commands use a dialog box instead of the message line prompt.

It is possible to drag files from the Windows File Manager onto MicroEMACS, providing a macro has been set-up to handle them.

MDI windows (aka screens) and the MicroEMACS frame window can be resized by dragging their border with the mouse or using the sizing buttons.

Text can be scrolled into view by using the scroll bars located at the right and bottom of each screen.

When MicroEMACS is running a macro, waiting for user input on the message line, or reading/writing a file, it is possible to input menu or other mouse commands, but only a subset of features is available. In particular, resizing is disabled and most menu options are grayed.

It is possible to terminate MicroEMACS at any time, using the "Close" (Alt+F4) item of the upper-left corner menu box. If there are modified buffers, or a file write operation is in progress, a confirmation is requested.

The amount of memory available for buffers is limited only by the actual (conventional and extended) memory available, including virtual memory when running Windows NT or Windows 3.x in 386-enhanced mode.

MicroEMACS can synchronize with other applications it launches.

MicroEMACS runs as a well-behaved Windows application, sharing the processor with other applications, even when a lengthy operation is in process.

Under Windows 3.x, MicroEMACS is a protected mode-only application: **it does not support real mode**, and runs only under standard or 386-enhanced mode.

The following page are distributed with MicroEMACS for Windows and loaded by the emacs.rc start-up file supplied in the distribution package:

CUA.CMD	Common User Access macros
DEV.CMD	example macro for software development
MDI.CMD	macros to map files to MDI windows

In addition, if a page named CUSTOM.CMD (to be supplied by the user) is found in the path, it is loaded after the three above.

371$ MS-Windows Specifics
372# Ms_WindowsSpecifics
373+ Index:2070
374K MS-Windows;windows

$375 #376 +377 K378 # CUA.CMD

This page is distributed with MicroEMACS for Windows and loaded by the emacs.rc start-up file. It contains a number of macros and rebinds many keys, in order make MicroEMACS more similar to other Windows applications that use the Common User Access standard.

To that end, a set of clipboard-related macros are supplied and you can select a piece of text by dragging the mouse across it while holding the left button held down or by moving around with the arrows or page keys while holding the Shift key down. That selection can then be **deleted** by pressing the Delete key, **copied** to the clipboard with the Ctrl+Insert keys, **cut** with Shift+Delete and **pasted** from the clipboard with Shift+Insert

Additionally, the following general purpose macros that work on the selection are supplied:

A-U	**CUA-case-upper**	converts all the selected text to upper case
A-L	**CUA-case-lower**	converts all the selected text to lower case
A-W	**CUA-count-words**	displays on the message line the number of words, characters and lines that compose the selected text
A-=	**CUA-flip-selection**	exchanges the point with the other end of the selection
A-^M	**CUA-select-region**	(Alt+Enter) makes the current region the current selection

$_{379}$ #$_{380}$ +$_{381}$ K$_{382}$ **DEV.CMD**

This sample page is distributed with MicroEMACS for Windows and loaded by the emacs.rc start-up file. It contains a few of macros that demonstrate how some features of the macro language can be used to facilitate software development:

The **run-makefile** macro is added to the Execute menu. It spawns a shell to run the command specified by the **%make** user variable and synchronizes with it. When the make process is finished, its output is displayed in a buffer named "**Results**".

A series of macros are added to the Help menu. They search a specific help file for a topic matching the word under the point.

379$ DEV.CMD
380# DEV.CMD
381+ Pages:DEVCMD
382K DEV.CMD

$_{383}$ #$_{384}$ +$_{385}$ K$_{386}$ **MDI.CMD**

This page is distributed with MicroEMACS for Windows and loaded by the emacs.rc start-up file. It contains macros that make it easier to associate each buffer with a separate screen (i.e. an MDI window). To that end:

The **open-file** macro replaces the find-file command in the File menu and in key bindings (^X^F). Instead of reusing the current screen, it creates a new screen to house each newly opened file.

The **rebuild-screens** macro, invoked from the Screen menu, associates a screen to each buffer.

The **kill-screen** macro (A-K) deletes a screen and the associated buffer.

MDI.CMD also contains the **drop-files** macro that handles drag and drop actions by invoking the **open-file** macro for each dropped file.

$387 #388 K389 # Sorry, no help available on this topic

You have attempted to get Help for a term that the Help system does not recognize.

Here are some other ways to find Help for individual terms:

Help Search

1) Choose the **Search** button (Alt+S) from the top of this Help window (just below the menu bar).
2) In the Help Search dialog box, under Search For, type in the term you want Help for. If the term is indexed in the Help, you will go to that term in the upper list box. If the term is not indexed, you will go to the closest lexical match instead.
3) Press Enter or choose the dialog's **Search** button. You will see a list of 1 or more Help topics in the Topics Found

Alternatively, within the Help Search list box, scroll through the list to find a specific topic, then press Enter or choose the **Go To** button to jump to that Help topic.

Help Index

1) Use the **Index** button (Alt+I) and then choose the category that best fits your query.
2) Then traverse Help links through the topics until you find what you are looking for. If it is documented in the Help system, you should be able to find it within 4 or 5 topics.

387$ No help available
388# NoHelpAvailable
389K default