

QNX4.0

VEDIT™ Text Editor



VEDIT™

***Multiple File Windowing
Text Editor***

VEDIT
Multiple File, Windowing Text Editor
For Program Development
And Word Processing
One of the VEDIT Family of Editors
Version 2.10

Manual Written By:
Theodore Green,
and Charles Scott

Programmed By:
Theodore Green & Thomas Burt

Greenview Data, Inc.
2773 Holyoke Lane
Ann Arbor, MI 48103
Telephone: (313) 996-1299
FAX: (313) 996-1308

Copyright (C) 1991 by Greenview Data, Inc. All rights reserved worldwide. No part of this publication may be reproduced, in any form or by any means, for any purpose without the express written permission of Greenview Data.

DISCLAIMER

Greenview Data, Inc. and the authors make no claims or warranties with respect to the contents or accuracy of this publication, or the product it describes, including any warranties of fitness or merchantability for a particular purpose. Any stated or expressed warranties are in lieu of all obligations or liability for any damages, whether special, indirect, or consequential, arising out of or in connection with the use of this publication or the product it describes. Furthermore, the right is reserved to make any changes to this publication without obligation to notify any person of such changes.

This edition: April, 1991

Cover Art: Robert Tinney

ACKNOWLEDGEMENTS

We would like to thank the following organizations and people for their assistance with this project.

Wayne Barrett, for donating many hours to editing and enhancing this manual and for his helpful feedback over many years.

Canon USA, for donating the model LBP-8 Mark III laser printer used in creating this manual.

Zenith Corporation, for donating the model ZCM-1490 VGA flat screen monitor used in writing this manual and for photography used in our advertisements and brochures.

Micro-Labs, Inc., for donating all models of their Ultimate VGA cards used in developing VEDIT and writing this manual.

Video Seven Inc, ATI Technologies Inc. and Hercules Computer Technology, Inc. for donating VGA cards used in development and testing.

Microsoft Corporation, Logitech USA and Mouse Systems for donating the "mice" used in development and testing.

This manual was created using Ventura Publisher from Xerox in conjunction with VEDIT PLUS. V-SPELL was used for spelling correction.

TRADEMARKS

VEDIT is a trademark of Greenview Data, Inc.

MS-DOS and OS/2 are trademarks of Microsoft.

IBM and IBM PC are trademarks of International Business Machines.

Ventura Publisher is a registered trademark of Ventura Software, Inc.

WordStar is a registered trademark of MicroPro International Corp.

QNX is a registered trademark of Quantum Software Systems Ltd.

UNIX is a registered trademark of AT&T.



Contents

Chapter 1

Introduction 1

- Welcome to VEDIT 1
- Main Features of VEDIT 2
- Ready To Use Macros 4
- System Requirements 6
- Using this Manual 6
- Notation 8

Chapter 2

Quick Tutorial 11

- Starting VEDIT 12
- Entering New Text 12
- Moving the Cursor 13
- Deleting Text 13
- Typing Beyond the Visible Screen Area 15
- Line and Column Numbers 15
- Quick Cursor Movement 16
- Undo 18
- Using Menus 19
- Answering a Prompt 20
- Prompt History 21
- Context Sensitive Help 21
- Printing Text 25
- Saving Your Work 26
- The Escape Menu 26

Exiting from VEDIT	27
That's It!	27

Chapter 3

Editing Guide 29

Starting VEDIT	29
Invocation Options	30
Setting VEDIT Options in your Environment	31
Exiting VEDIT	32
Backup Files	33
The Status Line	34
Using Menus	37
Responding to Prompts	39
Point and Shoot File Selection	41
On-Line Help	42
Mouse Support	44
Menu System	44
Quick Cursor Movement	45
Block Operations	46
Search/Replace Options	47
Notes on Mouse Operation	47
Keyboard Characters and Screen Display	47
Entering Control and Graphics Characters	48
How Control and Graphics Characters are Displayed	48
End of Line Handling	49
The <Tab> Key and Tab Characters	50
Lower and Upper Case Conversion	51
Key Emulation Modes	52
Other Keyboard Input Options (Technical)	53
Other Screen Display Issues (Technical)	53
Keystroke Macros	53
Scrolling the Screen	56
Vertical Scrolling	56
Horizontal Scrolling	56

Wrapping Long Lines on the Screen	57
Block Operations	58
Marking (selecting) a Block of Text	59
Copying / Moving text within your file	60
BLOCK COPY and BLOCK MOVE	61
The Text Registers	61
Columnar Blocks	63
Search and Replace	68
Regular Expressions	68
Short Form Pattern Matching	76
Word Processing Functions	82
Definition of “Word” and “Paragraph”	83
Indenting Text (Left Margin)	84
Word Wrap (Right Margin)	85
Formatting and Justifying Paragraphs	86
Editing Multiple Files	88
Editing Multiple Files Without Windows	89
Switching Between Files (Edit Buffers)	91
Exiting and Quitting Multiple Files	91
Simultaneous Editing with Windows	92
Copying Text From One File to Another	93
Edit Buffer and Text Register Differences	93
Windows	94
Windows and Edit Buffers	96
Deleting Windows	96
Zooming A Window	97
Editing One File in Two Windows	97

Chapter 4

Command Macros 99

Introduction	99
VEDIT Macro Language	99
Command Macros	100
Macro Language in Keystroke Macros	102
Auto-Startup & Auto-Execution	104
Automatic Startup	104

Auto-Execution	106
Print - Print Macro	107
Wildfile - Multi-file Processing	108
Compare - File Comparison macro	110
Sort - Mailing List Sort	113

Chapter 5

Menu Reference 115

Block Menu	116
Set Marker	117
Set Column Marker	118
Copy to Cursor	119
Move to Cursor	120
Copy to Register	121
Move to Register	122
Insert Register	123
Delete	124
Write to Disk	125
Remove Markers	125
Swap Ends	126
Edit Menu	127
Insert Mode	127
Overstrike Mode	127
Enter CTRL Char	128
Erase Line	129
Erase EOL	130
Center Line	130
Format Paragraph	131
Repeat	132
Lower/Upper case (Submenu)	133
File Menu	134
R/O File	134
New (same buffer)	135
Open (new buffer)	136
Buffer Switch	138
Insert File	140

Directory	140
Save As	141
Save and Continue	142
Exit (Save)	143
Quit (Abandon)	143
Goto Menu	145
Beginning of File	145
Home in Memory	146
Zend in Memory	147
End of File	148
Line #	148
Column #	149
Set Text Marker	149
Goto Text Marker	150
Matching () { } [] <>	151
Help Menu	152
General	153
Status Display	153
Edit Buffers	155
Text Registers	155
Misc Menu	156
ASCII Table	157
Shell	158
Run Program	159
Color Toggle	160
Load/Execute Macro	161
Execute Macro	163
Keyboard Layout (Submenu)	163
Add Macro	165
Record Macro	167
Change Layout	168
Display Layout	172
Print Layout	173
Load from File	174
Save to File	175

Print Menu	176
All	177
Block	177
Eject/Flush	178
Reset	179
Config (Submenu)	180
Search Menu	181
Search	182
Search Options	183
Searching within a block	185
Replace	187
Again	188
Config (Submenu)	189
Undo Menu	190
Edit	191
Line	191
Deletion	192
Reset	193
Window Menu	193
Create	194
Delete	196
Switch	197
Horiz. Split	
Vert. Split	197
Remove All	199
Zoom	199
Config Menu	199
Tab Stops	200
Word Processing (Submenu)	201
Programming (Submenu)	201
Emulation (Submenu)	202
File Handling (Submenu)	205
Misc (Submenu)	206
Save to Disk	208
Read from Disk	210

Escape Menu	210
Remove Block Markers	211
Exit (Save)	211
Quit (Abandon)	211

Chapter 6

Edit Function Reference 213

BACKSPACE	213
BLOCK COPY BLOCK MOVE	213
CANCEL	214
CURSOR UP	214
CURSOR DOWN	214
CURSOR RIGHT	214
CURSOR LEFT	214
DELETE	215
DEL PREV WORD	215
DEL NEXT WORD	215
ENTER CTRL	215
ERASE EOL	215
ERASE LINE	215
ESCAPE	215
FIND	215
FIND AGAIN	216
HELP	216
INDENT	216
INSERT TOGGLE	216
LINE BEGIN	216
LINE END	217
MENU	217
NEXT LINE	217
NEXT PARAGRAPH	217
NEXT TAB STOP	217
NEXT WORD	217
PAGE UP	217
PAGE DOWN	217
PREVIOUS PARAGRAPH	217

PREVIOUS WORD	217
REPEAT	217
REPLACE	218
RETURN	218
SCREEN BEGIN	218
SCREEN END	218
SCROLL UP	218
SCROLL DOWN	218
SCROLL RIGHT	218
SCROLL LEFT	218
TAB CHARACTER	219
T-REG COPY T-REG MOVE	219
T-REG INSERT	219
UNDENT	219
VISUAL ESCAPE	220
VISUALEXIT	220

Chapter 7

Configuration 221

Location of Configuration Files	221
What can be Changed via Configuration Files?	222
Format of Configuration Files	222
Configuration File, Line by Line	223

Chapter 8

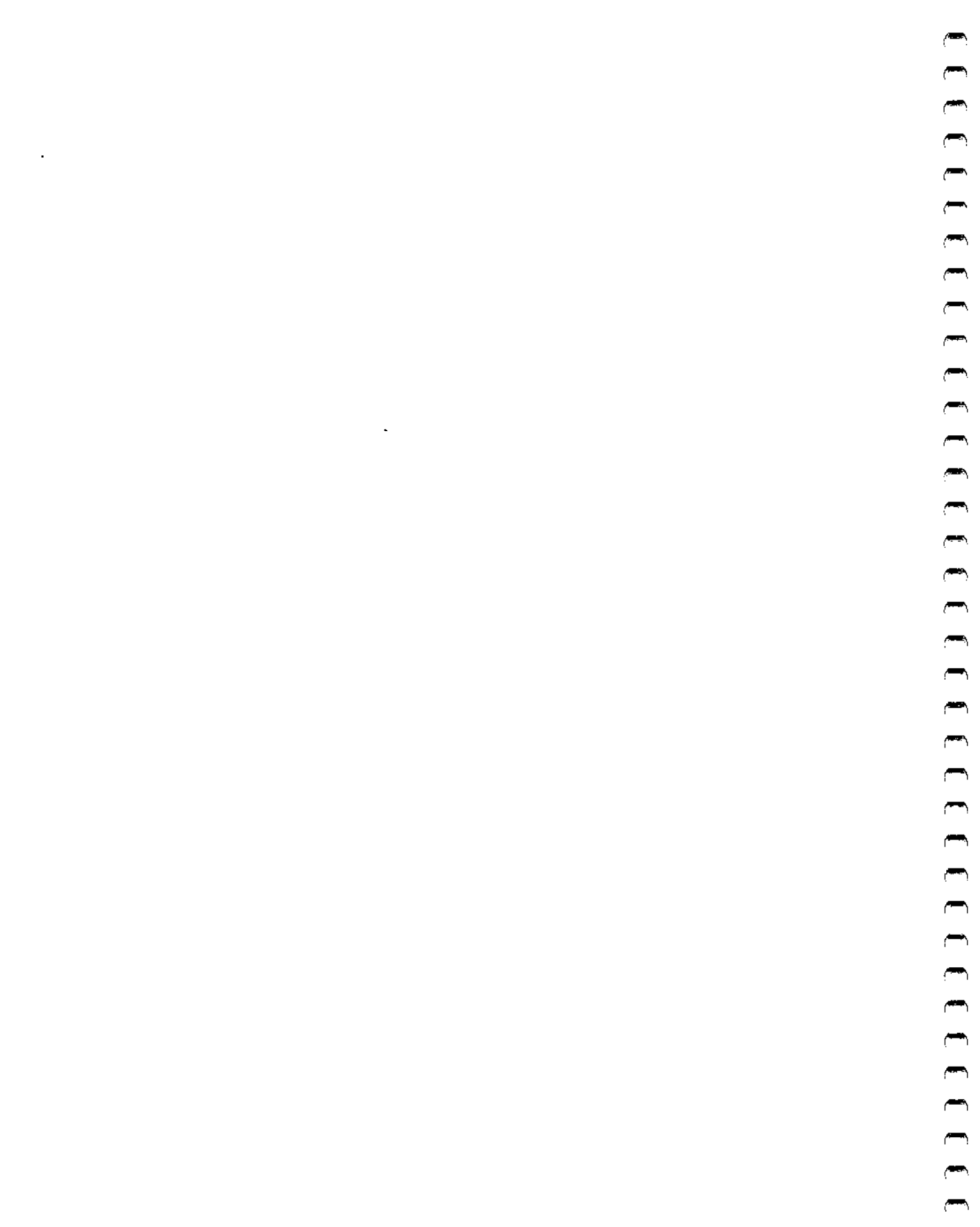
Messages 233

Appendices 245

File Management	245
Basic File Handling	245
Automatic File Buffering	246
Maximum File Size	247
File Locking	248
Memory Management	249
Memory Allocation	249
Operating with Limited Memory	250
UNDO Memory Management	251

Keyboard Layouts	252
VEDIT Default Keyboard Layout	252
Keystroke Macro Examples	256
Color Selection	256
Word Count	257

Index 259



Chapter 1

Introduction

Welcome to VEDIT

Purpose of Program	VEDIT is a full-screen text editor intended for both program development and document preparation. It is ideal for programming and for lengthy documents such as reports or manuscripts. It is also well suited to the preparation of text files for use with Desktop Publishing packages.
Simple, Yet Powerful	VEDIT is simple to use for the novice, yet has the speed, flexibility and power to satisfy the demanding professional. It offers a rich assortment of editing capabilities with complete online help facilities. It can be operated either with menus or with powerful keystroke commands.
Totally Configurable	VEDIT lets you completely determine your own keyboard layout and set other options to configure VEDIT to your particular hardware, applications and personal preferences. VEDIT for QNX comes set up for the recommended default keyboard layout and command set as described in this manual.
Emulations	With appropriate settings, VEDIT can closely emulate other editors and word processors. Multiple emulations are supplied with the QNX version of VEDIT. Two of them are qed which emulates the QNX 2.15 ed, and vi which emulates UNIX vi.

Emulations are invoked like any QNX command, simply by typing the name on the command line. Technically, they are links (see **1n** in the *Utility Guide*) to the VEDIT executable.



One way to learn VEDIT is by running the supplied tutorial with the command **vedit -x tutorial**. This covers VEDIT features in depth and is worth the one or two hours it will take to carefully work through it.

Another method is to bring up VEDIT emulating a known editor and play with it. **vi** users can do this with the command **vi** and access VEDIT's extensive online help with the <Alt>-<F1> key combination. Users familiar with QNX 2.15 **ed** would use the command **qed** and access help with <Alt>-<F1>.

Between the tutorial, emulations and online help, we are confident that you won't need this manual very much.

Main Features of VEDIT

File Handling

- Simultaneously edit up to thirty-seven files.
- Edit files of any size—automatic file buffering handles files too large to fit into memory.

User Interface

- Display each file in its own window or have different parts of one file in separate windows.
- Set the color for each window, create forms and menus within windows and much more. Any window can be zoomed to full screen.
- Pull-down menu system for easy access to most editing functions.
- Mouse support.
- Editing functions can also be accessed with “hot keys” which bypass the menu system—the menu system does not slow down proficient users.

- Context sensitive help. Pressing the Help key gives you precisely the information you need at the moment.
 - ASCII table. Pop up the ASCII table or insert any desired control, printer or graphic character into the text. IBM PC box drawing characters are easily accessed.
 - Escape to QNX, execute one or more commands, and return to VEDIT with your editing environment intact.
- Configurable
- Keyboard layout is completely configurable—assign any key to any function.
 - Unlimited keystroke macros. Any combination of keystrokes can be assigned to a single key. Allows single key access to menu functions and creation of custom editing functions.
 - Keystroke macros may be automatically set up when VEDIT is loaded. Groups of macros can be saved to and loaded from disk.
 - Default settings of all switches and parameters can be configured. Screen colors, display of tabs and line-ending characters, cursor and much more.
 - The help text is user changeable and expandable. Add your own help prompts and messages, even create online help for other programs you use.
- Editing Features
- Multi-level Undo. Up to hundreds of the most recent edit changes can be reversed either step by step or line by line. Deleted text can be re-inserted in its original position or anywhere else.
 - Automatic indenting—simplifies editing of structured programs and can also be useful for word processing.
 - Automatic checks for proper nesting of parentheses. For example, check “{” and “}” matching in C programs.
 - Flexible block operations. Process blocks of text on a character, line or column basis. Blocks may be moved or copied directly or the 36 Text Registers may be used as scratchpad buffers for extensive “cut and paste” operations. Blocks can be copied from one file to another.

- Powerful search and replace. Supports sophisticated pattern matching, regular expressions, reverse searching, selective and global replace. Previous search strings can be quickly accessed.
- Flexible printing with variable margins and page size. Print a block of text or the entire file with single, double or triple spacing. The supplied `print.vdm` macro prints files with convenient header (file name, date and page number).
- Horizontal scrolling for editing long lines—spreadsheets, program listings. Optionally, long lines can be wrapped onto multiple screen lines.
- Fully adjustable tab positions. The <tab> key can enter a tab character or, optionally, spaces to the next tab position.
- Word processing. Word wrap, paragraph formatting between adjustable margins, optional justification, printing. VEDIT does *not* enter any “special” control characters into the text.

Ready To Use Macros

VEDIT runs “macros” written in the VEDIT macro programming language. Numerous useful macros are supplied with VEDIT and can be used without needing to understand their internal operation.

This manual does not document the macro programming language. Users needing to develop their own macros should obtain the VEDIT PLUS documentation.

The supplied macros are menu driven and self prompting. Anyone can use them without knowing much else about VEDIT. These macros can also be set up to automatically execute when VEDIT is invoked.

The following macros are supplied with VEDIT:

- **Wildfile macro.** This multiple file processing macro lets you perform a search, search and replace, or run another macro on an entire group of files. The group of files may be specified using the wildcards “?” and “*”. (Wildfile performs a superset of the QNX **grep** command.)
- **Compare macro.** Displays the differences between two files on the screen and lets you merge/edit them in any way desired. After displaying a difference the macro can re-align the files and then continue looking for further differences. For example, after finding where revisions were made, you can copy blocks of text from one file to the other.
- **Sort macro.** This mailing list sort macro sorts a typical list of mailing addresses into alphabetical order.
- **Print macro.** This easy to use print formatting macro lets you print files with the filename, date and page number at the top of each page. Containing only a dozen commands, it also serves as a simple example of a useful macro. It could easily be extended for fancier print formatting.
- **Display macro.** This simple file display macro permits a computer user with no VEDIT experience to browse through a file.
- **Find Function macro.** Allows browsing C source code by selecting functions off a menu of function names.

VEDIT also comes with two libraries of additional macros. The “Command Macros” can be loaded from disk and executed at any time. The “Keystroke Macros” can be built into the editor and assigned to a “hot key”.

- **Command Macro library.** Includes box drawing, splitting/merging very large files, converting tabs to spaces and editing keystroke macros.
- **Keystroke Macro library.** Includes selecting window color from a color chart, duplicating characters and lines, moving the cursor by sentence, listing lines containing a text string, transposing characters, counting words and much more.

System Requirements

VEDIT requires:

- QNX version 4.0 or later.
- A console or terminal supported by QNX.
- Enough free memory to load one copy of the VEDIT program per machine (no matter how many users there are), plus at least 40K per user for data. This allows one moderate size text buffer per user.

Additional memory is useful in several ways. It allows more users. It improves VEDIT performance, particularly with large files, by allowing VEDIT to hold more of the file in memory. It also allows each user to do more, for example opening multiple buffers to edit several files at once or running other programs from within VEDIT.

VEDIT memory usage is configurable by the user. You can tell it to use more memory to maximize your VEDIT performance and flexibility, or you can place limits on VEDIT memory usage to preserve resources for other users and applications.

Using this Manual

This manual is organized into the following chapters:

Introduction (Chapter 1)

Introduces VEDIT and lists the main features and system requirements.

Quick Tutorial (Chapter 2)

This short tutorial familiarizes you with the basic aspects of using VEDIT including the menu system, online help, how to save the result and exit VEDIT.

Editing Guide (Chapter 3)

Covers the main topics of how to use VEDIT in general terms.

Command Macros (Chapter 4)

Describes how to use the “command macros” supplied with VEDIT.

Menu Reference (Chapter 5)

Describes each item in the Menu system in detail. It includes many step by step directions for using the features of VEDIT.

The **{CONFIG}** menu described here lets you configure VEDIT and fine-tune it to your personal preferences and applications.

Edit Function Reference (Chapter 6)

Describes each basic Edit Function in detail.

Configuration (Chapter 7)

Explains the configuration files which control VEDIT setup. Many configuration changes can be made from within VEDIT and simply saved to these files. A few of the less common types of change are made by editing the configuration files.

Messages (Chapter 8)

Lists and explains all error messages and common prompting messages.

Appendices

The following topics are included in the appendices.

- Technical descriptions of VEDIT’s file handling and memory management.
- Additional keyboard layout information.

Notation

<Ctrl>-< x >	A control character, such as <Ctrl>-<U>, which is typed by holding down the <Ctrl> (Control) key and typing the letter, in this case <U>.
<Alt>-< x >	A control character, such as <Alt>-<E>, which is typed by holding down the <Alt> (Alternate) key and typing the letter, in this case <E>. This type of control character is most frequently used to quickly access a particular submenu, in this case the {EDIT} submenu.
<key>	Any other individual key such as <Enter>, <Esc> or <F1>.
<Enter>	The “Enter” key. Also named “Return” or “CR” on some keyboards.
<F1>	The default key for accessing the Main Menu.
<Alt>-<F1>	The default key combination for the HELP function. The actual key is displayed on the right end of the status line.
FUNCTION	A basic editing function such as CURSOR UP or INDENT . Each edit function name is usually followed with the keypress corresponding to the default keyboard layout. However, you are free to modify the keyboard layout. The actual key assignments are displayed by the HELP function.
{MENU, Menu-Item} x	When describing use of the menu system, selections will be shown within brackets optionally followed by one or more <i>x</i> ’s to indicate variable parameters. The Main Menu is selected with the MENU function

(<F1>). For example, {**GOTO, Line #**} **71** means that you select “**Goto**” from the Main Menu, select “**Line #**” from the “**Goto**” submenu, and then enter the value “**71**” at the line number prompt.



Chapter 2

Quick Tutorial

Let's take a quick tour of VEDIT. In this chapter you will learn basic text editing, using the menu system, displaying help and, most importantly, how to save the results. In short, you will know all that's required to really use VEDIT!

Names of editing functions are shown in bold capitals, e.g. **HELP** or **CURSOR RIGHT**. Many functions can be invoked either via the menus or via keypress.

Names of menus are shown in curly brackets, e.g. **{FILE}**.

Menu choices or subfunctions are shown within the brackets in lowercase. For example, we use **{FILE, Directory}** to indicate selecting "Directory" from the **{FILE}** menu.

Keypresses are shown in angle brackets, for example **<→>** indicates pressing the right arrow key while **<Alt>-<F1>** indicates a two-key combination, holding the **<Alt>** key down while pressing **<F1>**.

To keep things simple, this quick tutorial often just lists which key to press to perform various editing operations.

However, the rest of this manual is oriented towards the names of the editing functions. This is the most general approach—often the same function can be performed either with a keystroke or via menus and the particular keys vary with system configuration.



Since this quick tutorial is specific on which keys to press, it assumes that you have selected the “Normal” keyboard layout.

Starting VEDIT

To start VEDIT and begin editing a file called **news.txt**, type:

```
vedit news.txt
```

Unless you already have a file called **news.txt** in the current directory, VEDIT will create this file for you. If the file does exist, VEDIT will load it and display the beginning of the file. The screen will appear as:

```
LINE: 1 COL: 1 ALT-F1=Help
```

The “status line” at the top of the screen indicates the name of the file being edited, the location of the cursor by line number and column number, and which key performs the **HELP** function.

The status line is also used to display prompts, error messages and the top line of the menu system.

The “cursor” is initially at the top left of the screen. The cursor indicates the exact location at which any edit changes, such as typing in new text, will be made.

Entering New Text

VEDIT starts up in “Insert Mode”. As you type, new characters are inserted at the cursor position. You can tell that VEDIT is in Insert Mode by the **INS** on the status line. If you don’t see **INS** on the status line, press <Ins> once (the **INSERT TOGGLE** function).

You can now type some text which will appear on the screen and will soon be made into a file. Type in the following text. Press the <Enter> key at the end of each line. You can make simple corrections while typing by pressing <backspace>.

Pressing the <Enter> key moves the cursor to the beginning of the next line.

```

sample.txt  LINE:  4 COL: 14      INS      ALT-F1=Help
Bach simply aued the professional musicians
who met or just observed him. Their descriptions
indicate that Bach was an amazing performer and
conductor as well as a composer.

```

When you press <Enter> while in Insert Mode, you are inserting a **newline** character. Every text line ends in a **newline** character.

The **newline** character is normally invisible, but like many things in VEDIT, you can configure it to be visible.

Moving the Cursor

You can move the cursor to another part of the text using the “Cursor Control” or “arrow” keys:

```

      <↑>
<←>      <→>
      <↓>

```

To move the cursor back to the first line, press <↑> until the cursor is back on the first line.

If you keep pressing <↑> the cursor will no longer move, because you’re at the beginning of the file. Similarly, if you press <↓> while on the last line, the cursor will not move any further because you are at the end of the file.

Pressing <←> while at the beginning of a line will move the cursor to the end of the previous line. Similarly, pressing <→> while at the end of a line will move the cursor to the beginning of the following line.

Normally, VEDIT only lets you position the cursor at real characters in the text and at the ends of lines. You normally cannot position the cursor on the screen where there is no text. This too is configurable.

Deleting Text

You can delete the character at the cursor by pressing .

When you do that, the character just after the cursor will be “drawn back” to the cursor position.

Move the cursor just before the word “the”:

Bach simply awed the professional musicians

To delete the word (and the extra space) press four times:

Bach simply awed professional musicians

You can also delete the character just before the cursor by pressing the <backspace> key. This is helpful when you want to change the word you just typed.

If you now press <backspace> five times, the line will be:

Bach simply professional musicians

We have already seen **newline** characters being inserted with the <Enter> key. While you can't see it, a **newline** really is a character. Just as or <backspace> delete visible characters, they will also delete a **newline**.

To show this, move the cursor to the "T" in the word "Their" and press <Enter> to split the line in two:

```
sample.txt  LINE:  3  COL:  1  INS  ALT-F1=Help
Bach simply awed the professional musicians
who met or just observed him.
Their descriptions
indicate that Bach was an amazing performer and
conductor as well as a composer.
```

As you can see, a **newline** has been inserted.

Now press <backspace> to delete the **newline** again:

```
sample.txt  LINE:  2  COL: 31  INS  ALT-F1=Help
Bach simply awed the professional musicians
who met or just observed him. Their descriptions
indicate that Bach was an amazing performer and
conductor as well as a composer.
```

Of course the key can also delete characters, and a **newline** is no exception.

Try the following key sequence:

<Enter> <←>

In short, when you're at the beginning of a line and want to append it to the previous line, press <backspace>.

When you're at the end of a line and want to append the following line, press .

You can delete an entire line by pressing <Ctrl>-<X> (the **ERASE LINE** function); any following lines will then move up.

You can delete text from the cursor position to the end of the line by pressing <Ctrl>-<Z> (the **ERASE EOL** function).

You can blank out the current line by first pressing <Home> and then <Ctrl>-<Z>.

This is different from <Ctrl>-<X> in that it changes a line of text to a blank line instead of erasing it entirely.

Typing Beyond the Visible Screen Area

Now let's see what happens when you type in more than a single screen full of text.

Move the cursor to the end of your current text and press <Enter>. Then repeatedly type a few characters and press <Enter> until you reach the bottom of the screen.

As you get near the bottom of the screen, VEDIT will automatically "scroll" the screen upward to make room for new text. In effect, you could keep typing forever.

Now try typing a very long line. As you get to the right side of the screen, just keep typing. VEDIT will automatically "scroll" the screen to the left to make room for the new text.

Line and Column Numbers

VEDIT tells you exactly where you are in your file by displaying the line and column number of the current cursor position on the status line. The current line number is located just to the right of **LINE:** and the current column number is just to the right of **COL:**.

In the previous example, VEDIT had scrolled the screen to the left. If you look at the status line in that example, you will notice that **COL:** had changed to “20:”, which tells you that the left edge of the screen is at column 20. The real column number is still displayed on the status line in its normal location.

Quick Cursor Movement

Using arrow keys to move the cursor one character or line at a time is fine if you don’t have far to go, but what about long distance jumps?

Fortunately the arrow keys are only some of the cursor control keys available. There are also keys which can move the cursor by words, paragraphs, screens and to the beginning or end of a line or screen. These are described below.

Other movements using the **{GOTO}** menu are also possible and are described later in this manual.

Moving by Words

To move the cursor forward an entire word, press:

<F6> NEXT WORD

To move the cursor backward an entire word, press:

<F5> PREVIOUS WORD

The **<F5>** and **<F6>** keys will always place the cursor on the first character of a word. If the cursor is not currently at a word, **<F6>** will move the cursor to the first character of the next word and **<F5>** to the first character of the previous word.

Moving to the Beginning or End of a Line

To move the cursor to the end of the current line, press:

<End> LINE END

If the cursor is already at the end of the current line when **<End>** is pressed, the cursor will move to the end of the next line.

To move the cursor to the beginning of the current line, press:

<Home> LINE BEGIN

If the cursor is already at the beginning of the current line when <Home> is pressed, the cursor will move to the beginning of the previous line.

From working with other editors and word processors, you may be used to the <Home> and <End> working differently than they do in VEDIT. Don't worry, VEDIT has many configuration options that let it closely emulate other editing styles you may already be used to.

Moving by Screens

Well, actually just less than a screen. VEDIT comes configured to "page" the screen up or down two lines less than the actual text screen size. This allows some overlap between screens of text, thus providing a visual reference for easier reading.

To move the cursor forward an entire screen page, press:

<PgDn> PAGE DOWN

To move the cursor backward an entire screen page, press:

<PgUp> PAGE UP

Even though these functions refer to "pages", they bear no relation to printed pages for a completed document. They are simply "screen" pages and their size depends mostly on the number of lines of text your computer displays.

Moving by Paragraphs

What is a paragraph? Since VEDIT is a "Plain Text" editor, it doesn't introduce unseen control characters to mark where one paragraph ends and the next begins. Instead, it considers a blank line as a break between paragraphs. Even when the first line of a paragraph is indented, if there isn't a blank line between it and the previous paragraph, VEDIT will think it's all one big paragraph.

To move the cursor forward an entire paragraph press:

<Ctrl>-<F6> NEXT PARAGRAPH

To move the cursor backward an entire paragraph press:

<Ctrl>-<F5> PREVIOUS PARAGRAPH

The <Ctrl>-<F6> and <Ctrl>-<F5> keys always place the cursor on the first character of a paragraph. If the cursor is not currently at a paragraph, pressing <Ctrl>-<F6> will move the cursor to the first character of the next paragraph and <Ctrl>-<F5> to the first character of the previous paragraph.

Undo

VEDIT offers you the luxury of changing your mind. Perhaps you changed some text, then decided it was better before the change. Rather than deleting and retyping, you can ask VEDIT to “undo” those changes.

To undo your last action press:

<Alt>-<F7>

Each press of <Alt>-<F7> undoes the next previous action. Actions which can be “undone” in this way include inserting or overwriting characters, deleting characters and cursor movements. Up to the last 1,000 of these actions can be undone by repeatedly pressing <Alt>-<F7>.

To try it, first delete a line of text by pressing <Ctrl>-<X>. Then press <Alt>-<F7> and the line comes back.

Go ahead and make some other changes—deleting, overwriting and inserting text. Repeatedly pressing <Alt>-<F7> will reverse these changes step by step.

You can also undo changes line by line by pressing <Ctrl>-<U>.

This is a quick way to undo all the changes you just made to a line, but didn’t intend to make.

For this tutorial you don’t want to repeatedly press <Ctrl>-<U> or you will soon end up with a blank screen because that is how you started out.



You cannot reverse the effect of an Undo function. There is no “redo” or “un-undo”.

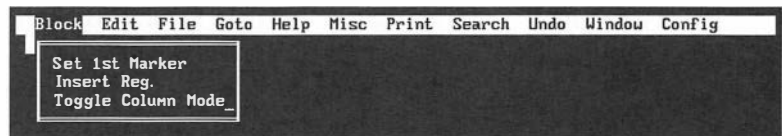
Using Menus

So far, you have learned how to perform basic operations by pressing particular keys which are assigned to them. Most of these operations are also available using menus, along with many that don't have a direct keypress equivalent.

To activate the menu system, press:

<F1> MENU

VEDIT will replace the status line with a horizontal menu bar. This is the "Main Menu" which shows the names of each "pull-down" menu. One item in the Main Menu will be highlighted and a pull-down menu will be displayed. VEDIT shows you this pull-down menu in advance of actually selecting it as a reference of what is available in it.



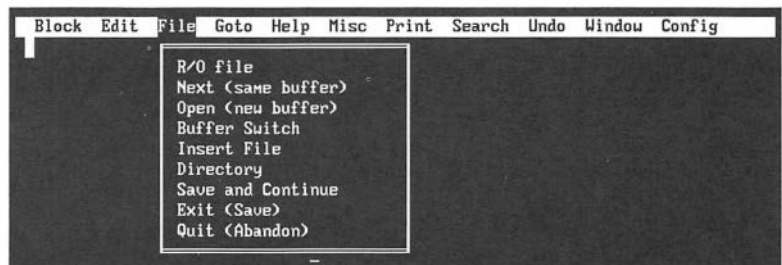
To highlight a different Main Menu item, press:

<→> or <←>

Try highlighting each item in the Main Menu and examine the items in each pull-down menu.

Before you can make a selection from a pull-down menu you must select it from the Main Menu.

Highlight "File" and press <Enter> to select the {FILE} menu.



You can tell that you have selected a pull-down menu by the highlight bar which appears in it. You can move this bar up and down to select the desired item.

To highlight a different menu item, press:

<↓> or <↑> or <spacebar>

Try highlighting each item in the **{FILE}** menu.

Items in the pull-down menus can also be selected by highlighting the desired one and pressing <Enter>.

In some cases these items will themselves be submenus. For example, in the **{MISC}** menu, the item **{MISC, Keyboard Layout}** is a submenu.

Answering a Prompt

The “Directory” item in the **{FILE}** menu lets you view the names of all files in your current directory or any other directory you specify. You can then press <Enter> to have VEDIT display the current directory or you can type in the directory of your choice.

To display the current directory, select “Directory” from the **{FILE}** menu; at the “Enter Filename:” prompt simply press <Enter>.

A typical display might be:

```
Press any key to continue_
Directory //22/user/fred/*
Makefile  functions.c  header.h  main.c
```

To display the contents of another directory, select **{FILE,Directory}** as before, then enter the directory name and an asterisk (*) to indicate that all files are to be displayed.

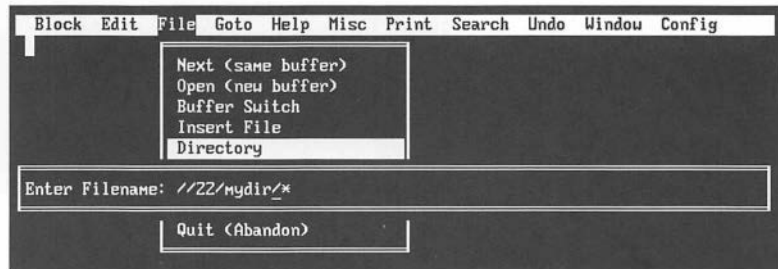
A typical display might be:

```
Press any key to continue_
Directory //Z2/mydir/*
Makefile  header.h  other.c  queries  to_do.list
doc.txt   main.c    problems support.c
```

Prompt History

VEDIT remembers what you enter in response to its prompts. You can recall and reuse these past responses or even edit them.

For example, to see this in action select **{FILE, Directory}** again. At the "Enter Filename" press <↑>. This will recall the previous directory name entry.



While editing a response, the <→>, <←>, <Home>, <End>, <Backspace> and keys may also be used.

Context Sensitive Help

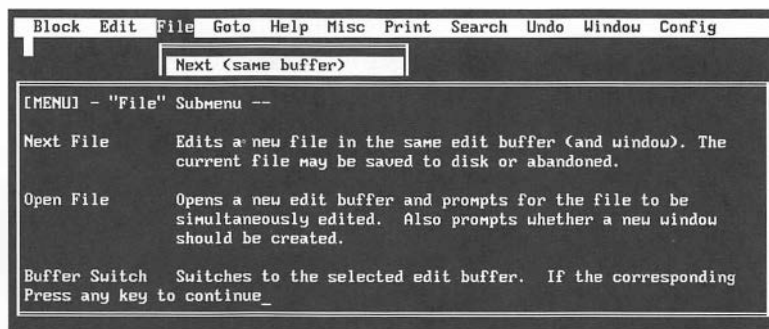
VEDIT provides *Context Sensitive On-Line Help*. When you press <Alt>-<F1> (the **HELP** function), VEDIT will "pop-up" additional information related to what you're currently doing.

- If the menu system is active, the information is on the current menu selection.
- If VEDIT is prompting you for something, the information is related to how you should respond.
- Otherwise, (i.e. if you are on a text screen without menus or prompts), the information describes the special keys available to you.



The exact Help information displayed may vary from system to system. For example, if your system administrator has modified keyboard layouts for local requirements or you have set up your own layout, then the information you get from Keyboard Help will not be as shown in our examples.

For example, activate the menu system and directly access the **{FILE}** menu by pressing <Alt>-<F>. Then press <Alt>-<F1> and you should see a screen like this:



The displayed help information may be more than one page long. You can advance to the next page by pressing any key. After the last page, **VEDIT** will return to the way it was before you asked for help.

Keyboard Help

VEDIT can also display a list of keystrokes and the operations they perform. To display keystroke help, make sure you are not in the menu system and press <Alt>-<F1>. The resulting screen should look like this:

LINE:	1	COL:	1	ALT-F1=Help
[BACKSPACE]	<---	[MENU]	F1	
[BACK TAB]	Home	[NEXT LINE]	CTRL-Enter	
[BLOCK COPY]	F9	[NEXT PARAGRAPH]	CTRL-PgDn	
[BLOCK MOVE]	ALT-F9	[NEXT WORD]	F6	
[CANCEL]	^C	[PAGE UP]	PgUp	
[CURSOR UP]	↑	[PAGE DOWN]	PgDn	
[CURSOR DOWN]	↓	[PREV PARAGRAPH]	CTRL-PgUp	
[CURSOR RIGHT]	→	[PREV WORD]	F5	
[CURSOR LEFT]	←	[REPEAT]	^R	
[DELETE]	Del	[REPLACE]	ALT-F2	
[DEL PREV WORD]	ALT-F5	[SCREEN TOGGLE]	^T	
[DEL NEXT WORD]	ALT-F6	[SCROLL UP]	CTRL-Home	
[ENTER CTRL]	^Q	[SCROLL DOWN]	CTRL-End	
[ERASE EOL]	^Z	[SCROLL RIGHT]	CTRL- →	
[ERASE LINE]	^X	[SCROLL LEFT]	CTRL- ←	
[ESCAPE]	Esc	[TAB CHARACTER]	---->	
[FIND]	F2	[TAB CURSOR]	<----	
[FIND AGAIN]	F3	[T-REG COPY]	F10	
[FORMAT PARAGRAPH]	^F	[T-REG MOVE]	ALT-F10	
[HELP]	ALT-F1	[UNDENT]	F7	
[INDENT]	F8	[VISUAL ESCAPE]	ALT-F8	
[INSERT MODE]	Ins	[VISUAL EXIT]	^E	
[LINE TOGGLE]	^L	[ZIP]	End	
Press key for desired function or <Enter> for next screen: _				

Each basic “Edit Function” is shown along with the keys assigned to it. These functions are described in detail in Chapter 6 (Edit Function Reference).

From this display, you can obtain additional information on a particular edit function by pressing the key(s) assigned to that function.

For example, pressing the key should display:

LINE:	1	COL:	1	ALT-F1=Help
[BACKSPACE]	Moves the cursor left and deletes the character there. At the beginning of a line it deletes the preceding <CR><LF> pair, effectively merging two lines together.			
[DELETE]	Deletes the character at the cursor position. The cursor doesn't move. A lone <CR> or <LF> will also be deleted, but a <CR><LF> pair will both be deleted as one.			
Press any key to continue_				

General Help

VEDIT can also provide help information on particular topics. First press <Alt>-<F1> then press <Enter>. Your display should look like this:

```
LINE: 1 COL: 1 ALT-F1=Help
===== BUILT-IN KEYSTROKE MACROS =====
{GOTO, Home}      {ALT-A}      {GOTO, Set}      {CTRL-S}
{GOTO, Zend}      {ALT-Z}      {GOTO, Goto}     {CTRL-G}

{CUNDO, Edit}     {ALT-F7}     {FILE, Buffer Switch} {F4}
{CUNDO, Line}     {CTRL-U}     {WINDOW, Switch}   {ALT-F4}

{GOTO, Matching <> } {ALT-F3}     {MISC, Execute Macro} {SHIFT-F8}
{CPRINT, Block}   {CTRL-P}     {MISC, Keystroke, Add} {CTRL-D}

Edit additional file {ALT-N}      Toggle between files {ALT-T}
Toggle EGA 43 line mode {ALT-L}      Toggle text colors  {ALT-O}
Create vertical window {ALT-U}

Box drawing characters assigned to <CTRL-F1> thru <CTRL-F10>:
1      2      3      4      5      6      7      8      9      10
|      -      r      l      L      I      T      L      |      |
Press key for desired function or <Enter> for next screen: _
```

This shows the keyboard macros currently defined for VEDIT. For now, don't attempt to understand these. Just press <Enter> again to see the next help screen:

```
LINE: 1 COL: 1 ALT-F1=Help
{STAT}      Status line messages
{MARG}      Setting left and right margins
{FORM}      Formatting paragraphs
{HZ}       Horizontal scrolling
{IC}       Inserting control and graphics characters
{CONU}     Lower / upper case key conversion
{EMPTY}    Emptying a text register (free more memory space)
{FILES}    Simultaneously editing multiple files
{COMWIN}   Creating a window for Command Mode
{EXIT}     Exit from Visual Mode to Command Mode

Type the word between "C" and "J" for help on above topics: _
```

This screen lists topics on which additional help information is available. To select a topic, simply enter its name.

Subnet (PRINT ALL) to print the entire file. After the test is full

Notice that the text started a few lines down on the page and was not printed on the left edge of the paper, but rather has about a one inch margin. The description of the **{PRINT, Config}** sub-menu in Chapter 5 (Menu Reference) explains how to configure various printing options.

Saving Your Work

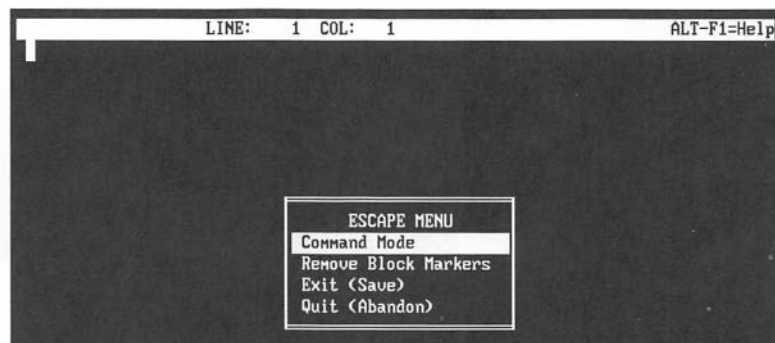
You should periodically save your work back to disk. You can do this using VEDIT's menu system. The following example also introduces you to two menu short cuts.

To save your work to disk, first press <Alt>-<F> to get the **{FILE}** menu. Then press <S> for "Save". There may be a short delay while your file is saved to disk.

In the above example you went directly to the **{FILE}** menu by pressing the <Alt> key in combination with the first letter of the Main Menu item. You can use this trick to go directly to any Main Menu item. The second short cut was to press the highlighted letter of the desired menu item; this is usually the first letter.

The Escape Menu

The **{ESCAPE}** menu is totally separate from the Main Menu. This menu is displayed when you press <Esc> and VEDIT doesn't know from what you want to "escape".



You select items from this menu using <↑> and <↓> just as you do from other menus.

Exiting from VEDIT

To end an edit session, simply press the <Esc> key (you may have to press it twice on a terminal) and select “Exit (save)” or “Quit (abandon)” from the resulting menu.

“Exit (save)” saves your work while “Quit (abandon)” leaves the editor without saving any text you may have created.

The usual way out of **VEDIT** is just <Esc> <E>. This two-key sequence saves your file and exits.

Note that selecting “Exit File” from the {**ESCAPE**} menu is identical to selecting “Exit File” from the {**FILE**} menu.

That's It!

That's all you need to create and edit text files. Obviously there's much more **VEDIT** can do for you. For example, **VEDIT** can copy or move blocks of text, search for and replace words or entire phrases and perform numerous other operations. Take some time to get comfortable with basic editing, then read on through the more detailed chapters which follow.



Chapter 3

Editing Guide

Starting VEDIT

Begin editing by typing an editor name optionally followed by one or two filenames. In the most common case the editor name will be “**vedit**” and only one filename will be given. For example:

```
vedit letter.txt
```

This command allows you to edit the file **letter.txt**.

If that file already exists it is read in at the start of the edit session and will be overwritten if an edited version is saved. Depending on VEDIT configuration, a backup of the original may also be saved in **letter.txt.b**.

If the file does not exist, the message “NEW FILE” will briefly appear. The file **letter.txt** will be created when you save your edited text.

In either case, the filename is displayed on the status line at the top of the screen to indicate which file you’re editing.

VEDIT can also be invoked with two file names:

```
vedit infile outfile
```

This lets you take an existing file *infile*, make changes to it and save it as the new file *outfile*. *infile* is not altered in the process.

Invocation Options

In addition to the file names, one or more “options” can be specified when you start up VEDIT.

- b** This puts VEDIT into *Browse Mode*. You can view files, but cannot alter them. You also cannot create files or write blocks of text to disk.
- c *command*** The VEDIT macro language command *command* is executed upon startup, *after* the configuration-specified startup files.

commands containing spaces or other characters meaningful to the shell must be quoted. Usually the simplest method is to enclose the entire command in single quotes (''). See the *Utilities Guide* **shell** section for other methods.
- i *execfile*** Execute **execfile** on startup *instead of* the normal **veditrc** file.
- m** Causes VEDIT to use screen attributes which are appropriate for a monochrome (Black and White) monitor.
- r** *Restricted* VEDIT. Only the file named on the command line can be edited and saved—no other load or save operations will be allowed. The user cannot list a directory from within the editor. Escape to the QNX shell and the ability to run QNX commands or VEDIT macro files are disabled.
- s *number*** Restricts VEDIT to use a maximum of *number* Kbytes of memory for all data areas. The minimum value is “40”. If the **-s** option is not given or is given without a *number*, the default limit is 192K.
- x *execfile*** Execute **execfile** on startup *after* the normal **veditrc** file.

Notes: Small values with the **-s** option may limit Text Register operations and the number of Undo levels available and reduce speed in editing large files. Large values may adversely affect other users and applications. See the Appendix section on “VEDIT Memory Management” for more detail if required.

All filenames may be any pathname QNX supports. Editing a file in another directory or on another network node requires only that you know how to name that file with a QNX pathname and that you have appropriate access permissions. See your QNX OS documentation for details.

See Also: “Exiting VEDIT” and “Backup Files” later in this chapter. “Auto-execution” in Chapter 4 (Command Macros). {MISC, Shell}, in Chapter 5 (Menu Reference).

Examples: To edit the file **myfile** in the current directory, type:

```
vedit myfile
```

After you have modified and saved **myfile**, you will have an updated copy called **myfile**. If VEDIT’s automatic backup feature has been enabled, you will also have the original copy renamed to **myfile.b**.

To edit the file **phone.list** and limit the data memory used by VEDIT to 40 KBytes, type:

```
vedit -s40 phone.list
```

Setting VEDIT Options in your Environment

The environment variable **VEDIT** is intended to hold VEDIT options which should be set for every VEDIT invocation by a particular user.

The variable would normally be set up in the user’s **.profile** shell initialization file and **exported** to later shells. See the *Utilities Guide* shell section for details on these mechanisms.

Any options stored in this environment variable are processed by VEDIT before those given on the command line.

For example, command:

```
VEDIT=-s384
```

will set the environment variable and cause any VEDIT you subsequently invoke from this shell to act as if you had given the **-s384** option on the VEDIT command line.

Another common use is something like:

```
VEDIT='-s40 -r'
```

A system administrator can use this to restrict a user's VEDIT activity. Some such setting would, for example, be appropriate for users using VEDIT to compose messages on a bulletin board system or to log simple reports from a factory floor.

Exiting VEDIT

VEDIT gives you a great deal of flexibility when exiting, especially when editing multiple files. You can save or abandon all files being edited or just one file at a time.

You can exit VEDIT with the functions “Exit (Save)”, and “Quit (Abandon)”. These functions are available in both the **{FILE}** and **{ESCAPE}** menus for convenience and are fully described in Chapter 5 (Menu Reference).

To exit VEDIT and save the current file(s):

1. Select either **{FILE, Exit}** or **{ESCAPE, Exit}**.

If you are editing just one file, this will save the file and exit VEDIT.

If you are editing multiple files, VEDIT will prompt you whether you want to save just the current file (and switch to the next file), or save all files and exit VEDIT.

To exit VEDIT and quit (abandon) the current file(s):

1. Select either **{FILE, Quit}** or **{ESCAPE, Quit}**.

If you are editing just one file, VEDIT will prompt you for confirmation to quit the file and exit.

If you are editing multiple files, VEDIT will prompt you whether you want to quit just the current file (and switch to another file), or quit all files and exit VEDIT.

The “Quit (Abandon)” function causes VEDIT to discard any changes made to the text since the last time you saved the file (e.g. with **{FILE, Save and Continue}**) and returns to QNX.

If you made any text changes, VEDIT prompts for confirmation before abandoning the file(s). This confirmation is your last chance of retaining these edit changes.

Notes: VEDIT checks to see if any changes have been made before exiting. If no changes were made since starting VEDIT or since the last time the file was saved, VEDIT will not bother to save the file again or ask for confirmation before quitting.

If the attempt to save the modified file fails, VEDIT displays an error message and aborts the request to exit. A save can fail for many reasons—for example because a disk is full, because you lack permission to write the file or because the file does not exist and you lack permission to create files in the directory. Often simply saving the file under another name (perhaps on a different disk or in a different directory) solves the problem.

{MISC, Shell} and **{MISC, Run Program}** let you execute QNX commands such as **rm** or **mv** which unlink filenames from a directory. Be careful not to unlink any filenames which start with the same name as the file(s) you are editing; VEDIT may be using these files for temporary storage.

See Also: “Starting VEDIT” earlier in this chapter. **{FILE, Exit}**, **{FILE, Quit}** in Chapter 5 (Menu Reference).

Backup Files

VEDIT can be configured so that when you edit and modify an existing file, it preserves the original version.

The modified version gets the original name. The original version is preserved under a filename constructed by adding a **.b** extension to the original name. Any existing **.b** file is overwritten.

When enabled, this VEDIT feature provides one generation of automatic backup—you always have both the current version and the next-to-last version on hand.

QNX commands can of course be used to provide other backup facilities—for example **cp** and **pax** are commonly used for backups and **rm * .b** would delete all VEDIT backup files.

By default, QNX VEDIT disables the VEDIT backup feature. It can be enabled as shown below.

To turn on backup files:

1. Set {**CONFIG, File Handling, Enable Backup**} to “On”.
2. To make this change permanent in your VEDIT setup, use {**CONFIG, Save to disk**}.

The Status Line

VEDIT provides a wide variety of information on the “Status Line” located at the top of the screen.

Additional status information is available by selecting the menu item {**HELP, Status Display**}.

The Status Line may include the following information:

#R	The “#” is followed by the name of the active edit buffer. It does not appear when the main edit buffer is active.
BLOCK	Indicates that both block markers have been set. The highlighted block can be processed with the { BLOCK } submenu, printed with { PRINT, Block } indented/undented with INDENT and UNDENT . The block markers can be removed by pressing CANCEL (<Ctrl>-<C>).

C-N-S	On the console the status of the Caps/Num/Scroll Lock is displayed on the status line with the single letters “C”, “N” and “S”. This is not shown for terminals.
COL: xxxx	Indicates which column of text the cursor is on. When the screen is scrolled sideways, the “COL” changes to the column number corresponding to the left side of the screen. E.g. “40:112” indicates the left screen edge is at column 40 and the cursor is in column 112.
Command Mode	Indicates that VEDIT is running a “Command Macro” (a macro written in the VEDIT macro language). Only the filename and edit buffer name are displayed on the status line in “Command Mode”.
DISK	The disk is full - This requires immediate attention since the file currently cannot be saved to disk. See “Exiting VEDIT” later in this chapter for more information.
filename	Indicates the name of the file you are currently editing. Use { HELP, Status Display } to display the full pathname of the file.
INS	“Insert” mode is on. If this is not displayed, VEDIT is in “Overstrike” mode.
LINE: xxxx	Indicates on what line of the file the cursor is. The message is reduced to “LI:” for line numbers greater than 65535. When the message is all in capital letters the entire file is memory. Otherwise the message is displayed as follows:
Line:	The beginning of the file is currently in memory, but the end of the file is on disk.
lINE:	The end of the file is currently in memory, but the beginning of the file has been written to disk.

line:	Only the middle of the file is in memory, the remainder is on disk.
MORE	Indicates that at least one Text Register is also an edit buffer. Note: the “TEXT” message has higher priority and usually appears in its place.
RM: xx	Right margin value—only displayed if the right margin is set.
TEXT	Indicates that you have text in at least one Text Register (which is not also an edit buffer). If you forget, use { HELP, Text Registers } to see which registers contain text or are edit buffers. The “BLOCK” and “1-END” messages have higher priority and may appear in its place.
1-END	Indicates that only one block marker is set. The block of text is highlighted as you move the cursor. The block marker can be removed by pressing CANCEL (<Ctrl>-<C>).

Examples: **news.txt LINE: 1 COL: 1 INS N ALT-F1=Help**

The above example indicates that the file **news.txt** is being edited, the cursor is at the beginning of the file, “Insert Mode” is on, “Numlock” is on and the **HELP** function is assigned to <Alt>-<F1>.

report.91.aug Line: 233 40: 62 1-END CN ALT-F1=Help

The above example shows that **report.91.aug** is being edited, only the first part of the file is in memory, the cursor is located at line 233 and column 62, the screen is scrolled 40 columns to the right (the leftmost displayed column is actually column 40), the first (and not the second) block marker is set, “CapsLock” and “Numlock” are on, and the **HELP** function is assigned to <Alt>-<F1>.

Using Menus

VEDIT's menu system provides convenient access to many editing functions which complement the basic "Edit Functions". It also lets you access most configurable parameters and additional help and reference information.

You can access the items in the Menu system in three ways:

- Bring up the Menu system by pressing **MENU** (<F1>). You can then navigate through the available functions with the cursor keys. Then press <Enter> to select the highlighted item. If the item is an editing function, it is then performed; if the item is a submenu, that submenu is selected.
- Directly access the desired submenu by pressing the corresponding "hot key". The default keyboard layout generally used on the console defines <Alt>-<x>, where "x" is the first letter of each main submenu. For example, <Alt>- accesses the {BLOCK} submenu. You can then select menu items in the normal fashion.
- Directly access the desired editing function by pressing its corresponding "hot-key". In this case the Menus are not displayed; the selected editing function is immediately performed. The Menu system displays the currently available "hot-key" for each menu item. As you learn them you will be able to edit faster. You can also add "hot-keys" for additional menu functions that you want direct access to.

The menu system is straightforward in operation and similar to other menu systems. However, there are several special features you should be aware of:

- <spacebar> highlights the next item in the main menu and submenus.
- You can also select by typing the highlighted letter of the desired menu item. This is often, but not always, the first letter of that item. This technique *must* be used when defining Keystroke Macros (i.e. "hot keys").

- When entering a submenu with **<Enter>** or the **<Alt>-<x>** technique, the initially highlighted item is the last selected item in that submenu.
- You can also enter submenus from the Main menu with **CURSOR DOWN** and **CURSOR UP** which then highlight the first and last items in the submenu, respectively.
- Even after you have entered a submenu, **CURSOR LEFT** and **CURSOR RIGHT** will pop you back to the main menu and select the previous/next submenu.



If you cannot read the menus, cannot tell which item is currently selected, or cannot tell which letter in each item is highlighted, your screen attributes are not set correctly. See “Screen Color Parameters” in Chapter 7 (Configuration).

To directly exit from any menu or parameter prompt, press **CANCEL** (**<Ctrl>-<C>**). The **CANCEL** function aborts any menu selection or prompt and returns the editor back to normal editing.

If you only want to back up to a previous menu level, press **<Esc>**. Pressing **<Esc>** from the Main Menu will exit the menu system and return you to normal editing.

VEDIT displays the special **{ESCAPE}** menu when **<Esc>** is pressed and there is no pending operation from which to “escape”. This “Escape” menu can be used to cancel block markers or to exit VEDIT.

Context sensitive help is available from within the menu system. While a menu is displayed, press **HELP** (**<Alt>-<F1>**). The information will contain a short description of each item in the current menu.

Notes:

When you highlight a menu item which is itself a submenu, VEDIT gives you a preview of the submenu without actually selecting it. This is a little different from some other menu systems.

There is some overlap between the basic “Edit Functions” and the editing functions found in the Menu system. For example, {**EDIT, Erase Line**} is the same as **ERASE LINE**, and {**SEARCH, Search**} is the same as **FIND**. This overlap helps new users and lets you access editing functions with a mouse.

As a convenience, the menu system remembers the most recent path taken. This way you can simply press **MENU** followed by two or more <Enter> s to select the same menu item again.

Examples: To select “Status Display”:

1. Press **HELP** (<Alt>-<F1>).
2. Highlight “Help” using the Cursor keys. Press <Enter>.
3. Highlight “Status Display” using the Cursor keys. Press <Enter>.

To change the screen color:

1. Press <Alt>-<M> as a shortcut to the {**MISC**} submenu.
2. Press “C” to perform “Color Toggle”.

To repeat the above operation:

1. Press **MENU** <Enter> <Enter>.

To exit VEDIT:

1. Press **ESCAPE** to access the {**ESCAPE**} menu.
2. Press “E” to select “Exit VEDIT”.

Responding to Prompts

When VEDIT needs to prompt you for additional information, such as a filename, it uses a pop-up “dialog” box. You can edit your response as you enter it.

VEDIT maintains a “history” of the responses you enter at each type of prompt. You can recall and reuse, or even edit, these previous responses.

The following edit keys may be used while entering a response:

CURSOR UP	Recall previous response.
CURSOR DOWN	Recall later response.
CURSOR LEFT	Move cursor one character to the left.
CURSOR RIGHT	Move cursor one character to the right.
LINE BEGIN	Move the cursor to the beginning of the response.
LINE END	Move the cursor to the end of the response.
DELETE	Erase the character under the cursor.
BACKSPACE	Erase the character to the left of the cursor.
ERASE LINE	Erase the current response.
ESCAPE	Cancel the operation for which the information was requested.

Notes: Responses are always entered in “Insert Mode”. It is not possible to change to Overstrike Mode.

After editing a line, it is not necessary to move the cursor to the end of the line before pressing <Enter>.

Some prompts request only a single character. You respond to these simply by pressing the desired character. Do not follow your response by pressing <Enter>.

Examples: To insert two files with similar names, say “**chapter7**” and “**chapter8**”:

Select {**FILE, Insert**} and type “**chapter7**” and <Enter>.

Select {**FILE, Insert**} again; press **CURSOR UP**; use **BACKSPACE** to alter the filename and then press <Enter>.

Point and Shoot File Selection

Since it is often difficult to remember exact filenames, VEDIT features “Point and Shoot” file selection.

For any editing function which requires a filename, such as **{FILE, New File}**, VEDIT will first prompt you for the filename:

Enter Filename:

If you know the exact filename you can simply enter it.

Alternatively, you can press <Enter> to have VEDIT give you a display of all files in the current directory.

Or you can enter a “*filespec*” using the “wildcard” characters “*” and “?” to have VEDIT only display selected files. The format of the “filespec” is identical to that used for the QNX **ls** command.

A typical “Point and Shoot” display might be:

```
Press any key to continue_
Directory //Z2/mydir/*
Makefile  header.h  other.c  queries  to_do.list
doc.txt   main.c    problems support.c
```

The top line displays the current “filespec” including node and drive number.

To select a file from the “Point and Shoot” window:

1. Use the cursor keys to highlight the desired file.
2. When there are many files, **PAGE UP** and **PAGE DOWN** may help in viewing the files; **LINE END** (<End>) highlights the last file and **LINE BEGIN** (<Home>) highlights the first file.
3. Press <Enter> to select the highlighted file.

4. Selecting the special filename “..” moves up in the directory structure. Any subdirectories are displayed before other filenames and are indicated with a following “/”. Selecting a subdirectory moves down in the directory structure to that subdirectory.

You can change the “filespec” from within the Point and Shoot window.

To change the Point and Shoot “filespec”:

1. Press <Tab>. The cursor will move just past the current filespec.
2. Edit the filespec as desired. Remember that you can use <↑> to recall previous filespecs.
3. Press <Enter> when the filespec is correct. VEDIT will display a new set of files with the first one highlighted.

Press **ESCAPE** or **CANCEL** (<Ctrl>-<C>) to cancel the Point and Shoot operation.

Notes: To access files on a different node you must change the “filespec”.

On-Line Help

Help information may be requested at any time by pressing **HELP**. The keystroke assigned to **HELP** is always displayed at the right end of the status line.

VEDIT’s help system is “Context Sensitive”, which means that it will display information which is relevant to the task at hand. For example, when you are performing simple type and edit operations, requesting help will display general information on key assignments and basic operations. If you request help while a menu or prompt is displayed, information on that particular menu or prompt will be provided.

The “General” help information is divided into three pages:

Keystrokes Assigned to Edit Functions

This is a list of all “Edit Functions” and the keys which are currently assigned to them. When this page is displayed, you can press a listed keystroke to display additional information on the associated Edit Function.

Keystrokes Assigned to Built-In Keystroke Macros

VEDIT’s default keyboard layout includes numerous “Built-In” Keystroke Macros. This page summarizes these macros and displays the keystrokes to which they are assigned. There is no additional online help available for these macros.

Help Topics

This page is a list of topics on which additional information is available. You can select the desired topic by entering the key word displayed between the “{ }” characters to the left.

Notes: Pressing **HELP** during normal editing (i.e. no menus or prompts are up) is the same as selecting **{HELP, General}**.

All help information, with the exception of Edit Function assignments, is contained in a file called **vhelp.hlp** which is normally in the directory **/usr/lib/vedit**.

The help system’s display of the Edit Function assignments is dynamic—it reflects any changes you may have made to the keyboard layout. However, the summary of the keystroke macros is static; this information is listed at the beginning of the **vhelp.hlp** file. It is only applicable to the default keyboard layout and will not be correct if you have an alternative keyboard layout. However, you can edit **vhelp.hlp** as desired.

A description of how VEDIT locates specific information in the **vhelp.hlp** file is beyond the scope of this manual. If you require this information, request a copy of “Adding Help to VEDIT” from technical support.

See Also: The **{HELP}** menu and **{MISC, Keyboard Layout}** submenu in the Chapter 5 (Menu Reference).

Mouse Support

The integrated mouse support makes it easy to navigate and select from the menu system. The mouse can also quickly scroll to other parts of the file, perform block operations, switch between windows and access online help.

If you have a mouse and it is active, the mouse cursor will by default be parked in the top right corner of the screen upon entering VEDIT. The cursor is invisible in that position but becomes visible as soon as you move it down into the screen.

{CONFIG, Misc, Mouse Cursor} controls the mouse support.

A value of “0” disables the mouse support. The default value of “1” starts the mouse cursor in the top right hand corner where it remains invisible until moved somewhere else. This is the preferred value for experienced users. The default value of “2” starts the mouse cursor in the middle of the screen. This reminds new users that VEDIT has mouse support.

The mouse support is fully described in the online help.

To access online help for mouse support:

1. Press **HELP** (<Alt>-<F1>).
2. At the prompt “Press key for desired function...” click the left mouse button.

OR

2. Press <Enter> twice to see the “Help Topics” and enter “mouse”.

Menu System

To access the menu system move the mouse to the status line and click the left button. To access the “Escape” menu click the right button.

Select submenus and items by moving the mouse to them and clicking the left button.

Quick Cursor Movement

To navigate the menu system hold down the left button. Sliding the mouse along the top screen line displays each submenu. Sliding the mouse up/down highlights each item or sub-submenu. Placing the mouse on an item and releasing the left button selects the item.

To escape from the menu system move the mouse off the menus (into the text) and release the left button. This is the same action as **CANCEL**.

Pressing both mouse buttons also performs a **CANCEL**.

The best way to do this is to press and hold the left button; then click the right button and release the left button.

Clicking the right button within the menu system accesses the online help.

The edges of the current window act as scroll areas for jumping to other parts of the text. These areas are used in conjunction with the right button.

R	SCROLL UP	PAGEUP
E		
L		
A	SCROLL LEFT	SCROLL RIGHT
T		
I		
V		
E	SCROLL DOWN	PAGEDOWN

To **PAGE DOWN** move the mouse to the six rightmost columns of the bottom line and click the right button.

To **PAGE UP** move the mouse to the six rightmost columns of the top line and click the right button.

To **SCROLL DOWN** or **SCROLL UP** move the mouse to the bottom or top window line and hold the right button as long as desired. Moving the mouse further to the right scrolls the screen faster.

To make quick jumps within the file, move the mouse to the left column and click the right button. The top left corner jumps to the beginning of the text; the bottom left jumps to end of the text. The lines in between correspond to relative positions in the text.

NOTE:



These jumps are relative to the text in memory and not necessarily to the entire file. It may be quicker to use the **{GOTO}** menu for jumping around in large files.

To **SCROLL LEFT** move the mouse anywhere on the left half of the window which is not in one of the defined areas above and click the right button.

To **SCROLL RIGHT** move the mouse anywhere on the right half of the window which is not in one of the defined areas above and click the right button.

Block Operations

To move the editing cursor to the mouse position, click the left button. Or you can drag the edit cursor by holding down the left button and moving the mouse.

Pressing the left button a second time (while the mouse is at the editing cursor) sets the first block marker. Either release the left button to set only the first marker, or hold down the left button and drag the editing cursor (with highlighting) to the position of the second marker. Releasing the left button then sets the second marker.

HINTS:



To highlight a small block move the mouse to the desired 1st position. Click the left button once and then press and hold the left button. Move the mouse to the 2nd position—the highlighting will follow the mouse. Release the left button.



To highlight a large block move the mouse to the desired 1st position. Click the left button twice. Move the mouse to the second position using any method. This may involve using the right button. Then click the left button.

To copy a block, highlight the block, setting both markers. Move the mouse to the position for inserting the block. While holding down the <Ctrl> key, click the left button.

Moving a block is similar, except that you hold down the <Alt> key and click the left button.

Search/Replace Options

The top line of the dialog box for the Search/Replace functions reads “Press [FIND] or [REPLACE] for Options”. To access the search options move the mouse to this top line and click the left button.

Notes on Mouse Operation

When normal prompts are on the screen, the left button acts identically to the <Enter> key. Therefore, the prompt “Press any key to continue” is satisfied by clicking the left button.

When normal prompts are on the screen, the right button acts identical to the **HELP** key and supplies context sensitive help.

Having both buttons depressed always performs a **CANCEL**, cancelling any current operation.

To make the mouse cursor invisible, move it to the right-most column of the status line. When you move the mouse out of the corner, the mouse cursor will reappear.

Keyboard Characters and Screen Display

During editing, all displayable characters such as letters, digits and punctuation are entered directly into the text, while function and control keys are interpreted as basic *Edit Functions* or as *Keystroke Macros*.

VEDIT allows complete flexibility in determining the desired keyboard layout. You can assign the basic Edit Functions to any function and control keys, and build in as many Keystroke Macros (“hot-keys”) as desired.

Function and control keys which are not assigned to either Edit Functions or Keystroke Macros are usually ignored.

Entering Control and Graphics Characters

There are 256 possible characters since a byte of storage can have 256 different values.

The first 32 characters are called “*control characters*” and have decimal values “0” through “31”.

The normal displayable characters have decimal values “32” through “127”.

The characters with values of “128” through “255” are called “*graphics characters*” or “*high bit characters*”.

To enter a control character into your text you must precede it with **ENTER CTRL** (<Ctrl>-<Q>). This technique also lets you enter control characters into search strings.

ENTER CTRL is fully described under the equivalent {**EDIT, Enter CTRL Char**} in Chapter 5 (Menu Reference).

Graphics characters can be entered directly by pressing and releasing the <Alt> key and then typing the two-digit hexadecimal value of the desired character.

Alternatively, you can enter control and graphics characters into your file using {**MISC, ASCII Table**}.

This is the only way to enter the null character with value 00.

How Control and Graphics Characters are Displayed

Most control characters are normally displayed as a “^” followed by the corresponding letter, e.g. <Ctrl>-<G> displays as “^G”.

Alternatively, control characters can be sent directly to the screen by changing the configuration parameter {**CONFIG, Misc, Screen display options**}. On the console (i.e. the “smiling face” on an IBM PC)

Graphics characters are normally displayed literally, but can alternatively be displayed as a decimal value in the format “<nnn>” by using the above configuration parameter.

<Ctrl>-<I>, the Tab character, is normally displayed as spaces to the next tab stop. Alternatively, the Tab character can be displayed with any other character with the configuration parameter {CONFIG, Misc, Tab-display character}.

<Ctrl>-<J>, the **linefeed** or **newline** character at the end of each line, is normally displayed as a space, effectively making it invisible. Alternatively, a visible **newline** character can be chosen with {CONFIG, Misc, Newline display}.

<Ctrl>-<M>, the Carriage-Return character, is displayed as "<CR>".

End of Line Handling

Different systems have different ways of marking end-of-line in text files.

QNX 4.0 uses a single character, the ASCII **linefeed**, hexadecimal 0A. This is the same method used by UNIX and by other Posix systems.

Earlier versions of QNX also used a single character, but it was a different character—the ASCII **record separator** (hex 1E).

DOS uses a pair of characters—**carriage return** (hex 0D) and **linefeed** (hex 0A)—to mark end-of-line.

The normal way to deal with these differences is to let the QNX file system handle them for you. When correctly set up the QNX 4.0 file system can access files from a DOS disk or an older QNX disk, translating to new QNX format "on the fly". Consult your system administration manual or your system administrator for details.

If that is done, VEDIT (and all other applications) will see all files accessed from any disk as QNX 4 files. There will be no need to make VEDIT itself deal with other formats.

It is also possible, however, to set VEDIT up to handle DOS format files directly. This is most often useful for files received over telecommunications links. Such a file is on a QNX disk partition so it will not be automatically translated by the file system, yet it may well be in DOS format.

The configuration parameter **{CONFIG, File Handling, “Newline” character(s)}** determines whether VEDIT expects QNX files or DOS files. It determines how **newline** characters are displayed, deleted and inserted.

When set for DOS newlines, VEDIT treats the CR/LF pair as a newline. The <Enter> key inserts both characters into the text, or **BACKSPACE** deletes them as a pair, and <CR> is displayed only when not followed by linefeed.

VEDIT does not automatically convert a DOS file to QNX or vice versa. However, this can be done with a search and replace operation.

The <Tab> Key and Tab Characters

The <tab> key is identical to <Ctrl>-<I> and is almost universally assigned to the **TAB CHARACTER** function.

In “Insert Mode” the <tab> key normally inserts a “Tab” character into the text. Tab characters are displayed using spaces up to the next tab stop, even though these spaces do not exist in the text. As a convenience, you do not have to be in Insert Mode to insert text on top of a Tab character. The Tab character will not be overwritten until you reach its last displayed position.

Because of the importance of the <tab> key and Tab characters, several configuration options are available:

- The tab stops are normally set to every fourth column, but can be changed with **{CONFIG, Tab Stops}**.

Tab stops can be set at any desired columns, e.g. at 7, 20, 30 and 73. Tab characters which occur past the last tab stop are displayed as control characters, i.e. “^I”.

- Instead of displaying Tab characters using spaces, you can select another character with **{CONFIG, Misc, Tab-display character}**.

A suitable value on a console is “07”. This can make it easier to distinguish between Tab characters and spaces in your file.

- VEDIT can be configured to insert multiple spaces (up to the next tab stop) instead of a Tab character when <tab> is pressed. Although this uses more disk space, it is useful in applications

where tab stops outside of VEDIT will not be the same as those inside. This option is also handy with FORTRAN and COBOL programs. This option is selected with **{CONFIG, Emulation, Expand <tab> key with spaces}**.

- The operation of the <tab> and <Enter> keys normally depend upon the “Insert Mode”. This can be changed with **{CONFIG, Emulation, Typewriter emulation mode}**.

Lower and Upper Case Conversion

This topic is primarily applicable to programmers, particularly assembly language programmers.

Several modes are available for converting between lower and upper case letters as they are typed on the keyboard. These modes are selected with the parameter **{CONFIG, Programming, Lower/upper case key conversion}**, which can take on three values:

- 0 No conversion takes place. This is the default.
- 1 All lower case letters are converted to upper case. This is similar to the “Caps Lock” on a keyboard.
- 2 Conditional conversion—lower case letters are converted to upper case only when the cursor is to left of the “Conditional conversion character”.
- 3 Similar to (2) except that characters are reversed instead of being forced to upper case.

Modes “2” and “3” are specifically designed for assembly language programmers that prefer having the Label, Opcode and Operand in upper case and the comment in upper and lower case.

In Mode “2” lower case letters are converted to upper case if they occur to the left of the “*conditional conversion character*”, typically “;”. To the right of the “;” they are not converted. In this mode an assembly language program can be entered or edited with all lower case letters and VEDIT will automatically convert the labels, opcodes and operands to upper case while leaving the comment fields alone. The “conditional conversion character” may be changed with **{CONFIG, Programming, Conditional conversion character}**.

Mode “3” is similar to Mode “2”; instead of converting lower case to upper case, it reverses the case of letters appearing before the “;”. This makes it easier to enter lower case strings into a program (hold down the shift key to enter lower case letters).

NOTE:



This upper/lower case conversion option does not affect any existing text; use {**EDIT, Lower/Upper case**} to convert existing text.

Key Emulation Modes

Not only can you assign the basic Edit Functions to any desired function or control keys, but you can change how the commonly used Edit Functions work. This lets you emulate other editors and word processors and/or fine tune VEDIT to your preferences.

The “emulation modes” can be changed with the {**CONFIG, Emulation**} submenu. They are fully described in Chapter 5 (Menu Reference).

The emulation modes include:

- Cursor positioning modes—control how the cursor keys move the cursor past the ends of lines.
- Typewriter emulation modes—control how the <tab> and <Enter> keys work in “Insert” and “Overstrike” modes.
- Line emulation mode—control how the **LINE BEGIN** and **LINE END** functions (<Home> and <End> keys) work when the cursor is already at the beginning/end of the line.
- Screen emulation mode—control how the **SCREEN BEGIN** and **SCREEN END** functions work.

A little experimentation is best for understanding these modes and deciding which you like best.

More sophisticated emulation can be performed with Keystroke macros.

Other Keyboard Input Options (Technical)

The parameter **{CONFIG, Emulation, Keyboard input options}** controls several rarely changed options:

- The “8th” bit can be enabled or stripped when reading the keyboard. You always want it enabled on the console. Note that this has no effect on 8 bit characters already in the text. Such characters are left unmodified when they are read from disk or written to disk.
- Unassigned function/control keys can enter their codes directly into the text. This is not normally desirable; it may be of use with some foreign language CRT terminals.
- The case of all letters can be flipped—e.g. typing “a” gives you “A” and typing “A” gives you “a”. (We are not exactly sure why you would want this, but VEDIT has always had this capability.)

Other Screen Display Issues (Technical)

On the console VEDIT produces its own cursor. Its appearance is user selectable. You can choose a solid block, a blinking block or the normal QNX underscore cursor. These options are configured with the **{CONFIG, Misc} submenu**.

VEDIT cannot change the appearance of the cursor on CRT terminals.

VEDIT will interrupt screen updates when you are performing rapid screen changes. Operations such as **PAGE DOWN** require updating the entire screen. If you press another **PAGE DOWN** while the screen is updating, VEDIT interrupts the unwanted update and restarts to display the most current screen. You are most likely to notice this if you hold down the **PAGE DOWN** key.

Keystroke Macros

In order to use VEDIT effectively and streamline your own editing tasks it is important to understand VEDIT’s extensive “*Keystroke Macro*” facilities.

VEDIT can remember a sequence of keystrokes and repeatedly play them back at the press of a key. These stored keystroke sequences are called “Keystroke Macros”. They let you assign a frequently typed

sequence of keys to a single function or control key. You can define a virtually unlimited number of Keystroke Macros and each may contain up to several hundred keystrokes.

Each Keystroke Macro is normally assigned to a function or control key that is not already in use (examples might be <Ctrl>-<F5>, <Ctrl>-<W> and <Alt>-<J>). Pressing the single function or control key then performs the equivalent of typing the entire sequence of keys. This saves you time and effort and reduces the chance of error.

You can use Keystroke Macros to repeatedly execute almost any operation or sequence of operations. This includes selecting items from the menus, responding to VEDIT prompts as well as simply typing text.

Keystroke macros have several purposes:

- The most common use of Keystroke Macros is as “hot-keys” for directly accessing items within the Menu System. Many such “hot-keys” are built into the default and other supplied keyboard layouts. The menus display any “hot-keys” which directly access each item.
- Keystroke macros can access the VEDIT macro language; these keystroke macros always begin with **VISUAL EXIT**.

Although this manual does not describe the VEDIT macro language in detail, the appendices list useful Keystroke Macros including those built into the supplied keyboard layouts. The supplied file **keymacro.lib** lists other useful macros.

- Since new Keystroke Macros can easily be added at any time, you can define one whenever you find yourself typing the same sequence over and over again. For example, you could define the key <Ctrl>-<A> to type out the whole phrase “attached and included herein by reference” each time it was pressed.

You can add new Keystroke Macros at any time.

{MISC, Keyboard Layout, Add Keystroke Macro} define a new Keystroke Macro and edit it as you enter it. Alternatively, **{MISC, Keyboard Layout, Record Macro}** lets you record a new Keystroke Macro while you edit your file. See these functions in Chapter 5 (Menu Reference) for more details.

VEDIT forgets new Keystroke Macros when you exit unless you make them permanent with **{CONFIG, Save to disk}**.

Example - Adding a new
Keystroke Macro

In this example, you will define <Ctrl>-<F1> to enable justification. (You could also define <Ctrl>-<F2> to disable it.)

1. Select **{MISC, Keyboard Layout, Record Macro}** (or press <Alt>-<K>). You will see the following dialog box:

```
Press <Backspace> to edit; <Esc> to enter

Function/Control Key?
```

2. At the prompt, press <Ctrl>-<F1>.

If you make a mistake, press **BACKSPACE** to delete the keystroke. Now press <Esc> to accept <Ctrl>-<F1>.

You are now in “Record Mode” as displayed on the status line.

3. Justification is enabled by setting **{CONFIG, Word Processing, Justify Paragraphs}** to “1”. Therefore, press the following sequence of keys: (assumes <F1> is **MENU** and <Ctrl>-<C> is **CANCEL**)

```
<F1> C W J 1 <Enter> <Ctrl>-<C>
```

Turn off the Record Mode by pressing <Alt>-<K>. (The key to turn off (stop) Record Mode is displayed on the status line.)

The new Keystroke Macro is now fully defined.

The function **{MISC, Keyboard Layout, Display Layout}** is used to display all active Keystroke Macros. The <Ctrl>-<F1> defined in the example above would display as:

CTRL-F1 [MENU] CWJ1 [RETURN] [CANCEL]

Note that VEDIT records the Edit Functions in a Keystroke Macro and not the function and control keys that were pressed. This ensures that Keystroke Macros are independent of the keyboard layout—you can change the keyboard layout without having to reenter existing Keystroke Macros.

As a special case, our example Keystroke Macro must end in CANCEL (<Ctrl>-<C>) to exit from the menu system.

Scrolling the Screen

The screen can be thought of as a “window” into a much larger area of text. Certainly this larger area can extend above and below the currently visible text. However, if the text is logically wider than the actual screen width, it can also extend to the right or left of the visible text. The areas of text which are not currently visible can be exposed by “*scrolling*” the display over to those areas. Scrolling happens naturally when entering new text as the older text “scrolls” off the screen.

Vertical Scrolling

The screen scrolls automatically as the cursor is moved towards the very top or bottom of the screen. As delivered, VEDIT will scroll when the cursor reaches about the third line from the top or bottom of the screen. This ensures that you always see a few lines before and after the line you are editing.

You can also use **SCROLL UP** (<Ctrl>-<Home>) and **SCROLL DOWN** (<Ctrl>-<End>) to scroll the screen without having to move the cursor. This allows you to view lines which are just off the screen.

Editing the VEDIT configuration files (see Chapter 7) lets you select how many lines of overlap you will see when using **PAGE UP** and **PAGE DOWN**, and how close to the top/bottom of the screen the cursor can get before the screen scrolls.

Horizontal Scrolling

VEDIT can scroll the the screen sideways for editing documents which are wider than the display, such as spreadsheets or structured programs. Similar to vertical scrolling, the screen automatically

scrolls as the cursor is moved toward the beginning or end of long lines. You can also scroll the screen horizontally without moving the cursor using **SCROLL RIGHT** (<Ctrl>-<→>) and **SCROLL LEFT** (<Ctrl>-<←>).

When scrolling sideways, the screen normally jumps by 20 columns at a time. If desired, this can be changed with **{CONFIG, Emulation, Horizontal scroll increment}**.

When editing extremely long lines, you can quickly move the cursor to any desired column with **{GOTO, Column #}**.

Wrapping Long Lines on the Screen

Long lines normally extend off the right side of the screen. Alternatively, you can have them wrap onto multiple screen lines. This lets you see an entire long line at once.

To wrap long text lines onto multiple screen lines:

1. Select **{CONFIG, Emulation, Horizontal scroll margin}**.
2. At the prompt enter a new value of “79”.

The “*horizontal scroll margin*” (independent of the word processing right margin) determines at which column VEDIT wraps long lines.

Lines longer than the horizontal scroll margin are wrapped to the next screen (window) line. These additional screen lines are called “*continuation lines*” and are indicated with a special “*continuation character*”, typically a reverse video “-” (dash) in the leftmost column. The continuation character can be changed by editing configuration files; see Chapter 7.

When the “Horizontal scroll margin” is set, the leftmost column is reserved for the continuation character—lines which are not continuation lines have a space in the first column. Therefore only 79 characters can be displayed per screen line (assuming 80 character width); that is why “79” and NOT “80” is typically chosen for the Horizontal scroll margin.

The Horizontal scroll margin can be set to values up to 255 which is obviously wider than your screen. In this case, only lines longer than the scroll margin are wrapped. Normal horizontal scrolling is used to

view columns up to the scroll margin. Since continuation lines are quite obvious on the screen, this lets you flag lines longer than a specified length, e.g. lines that are too long for your application.

For example, your compiler may not support lines longer than 132 characters. In this case set the “Horizontal scroll margin” to 132 and any longer lines will be obvious on the screen.

Notes: Setting the Horizontal scroll margin slows down screen updates by about 25% due to this feature’s computational overhead.

To reduce the amount of unwanted side to side scrolling, especially on CRT terminals, the **LINE END** function can be configured to only go to the end of the screen line instead of the end of the text line.

Block Operations

One of the most useful features of VEDIT is its ability to copy, move, delete, save and reuse blocks of text.

A “*block*” can be any amount of text from one character to an entire file. A block can be either a group of contiguous characters called a “*character block*” or a vertical column which extends down through the middle of your text called a “*columnar block*”.

A block is selected (marked) by setting beginning and ending “*block markers*”. VEDIT highlights this area so you can see what you have selected.

Block editing functions such as {**BLOCK, Delete**} are not available, and are not even listed in the menus, until you have selected a block of text.

Alternatively, to move the block, i.e. “cut” it from its original position and “paste” it into the new position, select {**BLOCK, Move to Cursor**} or press **BLOCK MOVE** (<Alt>-<F10>).

Marking (selecting) a Block of Text

Selecting a block of text not only lets you perform block editing functions, but also changes the operation of other editing functions. For example, when the cursor is within a highlighted block of text, the **INDENT** and **UNDENT** functions indent/undent the entire block; otherwise these functions only change the current left margin.

The description for {**BLOCK, Set Marker**} in Chapter 5 (Menu Reference) describes in detail how to mark a block of text. In short, move the cursor to the beginning of the block and select {**BLOCK, Set Marker**} (<F9>) to highlight a “character” block, or {**BLOCK, Set Column Marker**} (<Ctrl>-<N>) to highlight a “columnar” block. As you now move the cursor, intervening text will be highlighted.

Many block operations including {**BLOCK, Delete**} and {**PRINT, Block**} can be performed after setting only the first marker—the cursor position acts as the second marker. However, if you need to or want to, you can set the second block marker by again selecting {**BLOCK, Set Marker**} or {**BLOCK, Set Column Marker**}.

After setting the second block marker, the status line will display “**BLOCK**” and moving the cursor will not affect the highlighted block. All block operations are now available.

You can also use the mouse to mark and highlight a block of text.



The default VEDIT keyboard layout assigns <F9> as a “hot-key” for {**BLOCK, Set Marker**}.

If your layout does not have an assignment for this function, you can alternatively press **BLOCK COPY** even if you don’t want to copy a block.

BLOCK COPY also sets the block markers—until both block markers are set, it only sets the block markers; when both block markers are set, it copies the block to the current cursor position.



To remove the block markers it is usually simplest to press **CANCEL** (<Ctrl>-<C>).

What exactly does the block include?

For “character” blocks you must mark the end of the block with the cursor one character **PAST** the last character to be included in the block. In other words, the character just before the cursor is the last character included in the block.

Therefore, if you end a “character” block at the end of a line, the (invisible) **newline** character will not be included. On the other hand, if you end the block at the beginning of the next line, the preceding **newline** character will be included.

This exclusion of the character at the cursor position may not make sense at first, but once you are used to it, it simplifies block operations.

Although VEDIT displays the “cursor” on top of characters, it is best to think of the cursor as being between characters. Indeed, “graphics mode” word processors such as Microsoft Word (tm) and Xerox Ventura Publisher (tm) display the “cursor” between characters. However, because this is not possible in “text mode”, VEDIT is forced to display the cursor on top of the next character.

If you think of the cursor as really being between characters, you will understand why VEDIT marks the ends of blocks the way it does.

However, we violate the above reasoning when marking “columnar” blocks. It is much more intuitive and useful to include the character at the cursor position when marking the end of a columnar block.

Copying / Moving text within your file

The description for {**BLOCK, Copy to Cursor**} in Chapter 5 (Menu Reference) describes in detail how to copy a block of text of text within your file.

In short, first select the block of text to be copied by setting both block markers. Next move the cursor to the position where you would like the same text. Then select {**BLOCK, Copy to Cursor**} or press **BLOCK COPY** (<Ctrl>-<F10>).

Copying a block of text to two or more places in your file or into another file is best done using “Text Registers”.

**BLOCK COPY
and BLOCK
MOVE**

BLOCK COPY and **BLOCK MOVE** are “smart” Edit Functions that give you additional flexibility in setting block markers. Their operation depends upon how many block markers are set—see the table below. The best way to understand their action is to try them in each situation.

For example, when no block markers are set, pressing **BLOCK COPY** sets the first block marker; pressing it again (after moving the cursor) sets the second block marker; pressing it a third time copies the highlighted block to the current cursor position. In this way, only a single key is needed to copy a block of text.

Block Function Key Behavior

Function	Key	No Marker Set	1st Marker Set	2nd Marker Set
BLOCK COPY	<Ctrl-F9>	Set 1st Marker	Set 2nd Marker	Copy to cursor
BLOCK MOVE	<Alt-F9>	Set 1st Marker	Set 2nd Marker	Move to cursor

Notes: The default VEDIT keyboard layout assigns <F9> to **{BLOCK, Set Marker}**.

Alternatively you could assign <F9> to **BLOCK COPY**. <F9> would then perform a convenient combination of setting block markers and copying a block of text. (Earlier version of VEDIT did this.)

**The Text
Registers**

VEDIT provides 36 text holding areas called “*Text Registers*”, which are similar to “memory” on a calculator. Using these registers, you can store text independently of the text you’re editing.

To copy a block of text into a Text Register, highlight the block of text and select **{BLOCK, Copy to Register}** or press **T-REG COPY** (<Ctrl>-<F10>). Alternatively, to also remove the highlighted block from the file, select **{BLOCK, Move to Register}** or press **T-REG MOVE** (<Alt>-<F10>).

You will be prompted for the register’s single character name. Valid register names are “0” through “9” or “A” through “Z”. Type one of these keys to select a Text Register. The status line message “TEXT” now indicates that something is stored in a Text Register.

You can insert the contents of a Text Register anywhere in your file or in another file. Place the cursor at the desired location and select **{BLOCK, Insert Register}** or press **T-REG COPY** (<F10>).

Text Register Usage

The 36 Text Registers serve three primary purposes:

- For “cut and paste” operations, where they temporarily hold a block of text.
- To hold sequences of commands in the VEDIT macro language which may be executed as “Command Macros”.
- As additional edit buffers for simultaneously editing several files.

In all cases, the registers are holding textual material; only the manner in which the text is used is different.

With 36 Text Registers available, it is easy to forget which register contains what. Although you could use many organizational schemes, we recommend the following scheme for using registers:

- | | |
|--------------|--|
| 0 | This is the default “cut and paste” register, since it is the easiest to use with the various block functions. |
| 1 - 9 | Are used as additional “cut and paste” registers or other general purpose registers. |
| A - F | Are used or reserved as edit buffers, for editing additional files. |
| G - Y | Are used for Command Macros. When you are not running Command Macros, these can also be used as edit buffers. |
| Z | This register is used by the auto-startup veditrc and any auto-execution Command Macros. |

One reason we recommend this scheme is that the **{FILE, Open File}** function creates new edit buffers beginning with “A”. (It uses the next empty register beginning with “A”.) Similarly, the **{WINDOW}** split functions create multiple windows per edit buffer beginning with window “0” so that these window names are unlikely to interfere with the names of edit buffers.

Notes: The description for {**BLOCK, Copy to Register**} in Chapter 5 (Menu Reference) covers many more details including how to append a block to any existing contents in the register.

The topic “Multiple File Editing” later in this chapter fully describes the differences between Text Registers and Edit Buffers.

All Text Register contents are lost when you exit VEDIT.

Command Macros have the ability to “write-protect” Text Registers used by the macro so that you don’t inadvertently alter them.

Columnar Blocks

VEDIT can manipulate columns of text. A “*columnar block*” is a rectangle of characters in the middle of your text. It can be anywhere from 1 character wide to the full width of the text being edited. It can also extend from as little as one line to many pages in length. Once you have marked (highlighted) a columnar block, you can perform any of VEDIT’s block operations on it. You can also restrict a search/replace operation to the columnar block.

Columnar blocks are especially convenient for editing tabular data. For example, you could copy a particular table of numbers, say between lines 10 and 20 and between columns 30 and 40 to a Text Register. You could then insert this table next to another table of numbers, say beginning on line 50 and column 60.

To make columnar block operations appear as natural as possible, VEDIT performs some additional manipulations on your text. First, it assumes that a columnar block being copied should have a flush right margin. Since some lines being copied may be too short to reach the block’s right margin, the copy of these lines is padded with spaces to make it flush. This insures that when the columnar block is inserted, it does not destroy the alignment of the following text. Similarly, when inserting a columnar block, spaces are added to pad short text lines which do not reach the insertion column. This keeps the inserted text aligned.

In some cases, such as inserting a columnar block at the end of text lines, these extra spaces become trailing spaces which are not needed for alignment purposes.

You can select with

{CONFIG, Emulation, Trim spaces after columnar block insert}
whether these trailing spaces are trimmed from the text.

Second, if the columnar block being copied contains Tab characters, these tabs are converted to spaces. (This is essential for columnar operations to work as expected.) When the columnar block is inserted, you can select with

{CONFIG, Emulation, Retab after columnar block insert}
 whether these spaces (and adjacent spaces in the existing text) are converted back to Tab characters.



This trimming and tab/space conversion applies only to the inserted text and adjacent spaces. It DOES NOT affect the entire file and only applies in COLUMN mode.

Columnar Block Examples

To perform columnar block operations mark the desired block with **{BLOCK, Set Column Marker}** (<Ctrl>-<N>). After setting the first block marker, the highlighting shows precisely which characters are included in the block. You should immediately notice the difference in the way the text is highlighted in Column Mode.



When columnar markers are set, **CURSOR RIGHT** will move past the end of short lines (similar to cursor positioning mode 4). This lets you set the right column past short lines.

The following screen shows a columnar block highlighted. Note that the cursor is in the lower right corner of the block. (Unlike character blocks, columnar blocks include the character at the cursor position.)

[illegible]

After copying this block to a Text Register, it will contain:

```
000000000
000000000
000000000
00000000.
000000...
0000.....
```

The periods indicate above indicate spaces which the columnar block copy added to the register in order to give it a flush right margin.

Now consider the following text before inserting this register.

[illegible]

After inserting the register, the screen will display:

[illegible]

This shows the importance of the added extra spaces when the block was initially copied to the Text Register.

Now consider the following text before inserting the same Text Register. Note that text lines 6 and 7 do not reach the insertion column.

[illegible]

After inserting the same register, the screen will display:

[illegible]

By padding text lines 6 and 7 with spaces, the inserted block also remains aligned.

Notice that the extra spaces added to the Text Register are now trailing spaces on lines 6 and 7 above and serve no alignment purpose. When **{CONFIG, Emulation, Trim spaces after columnar block insert}** is enabled (the default), these extra spaces are trimmed following the insertion.

To demonstrate this trimming, consider the following Text Register. Note that it contains a blank line consisting of spaces.

```
000000000
000000000
.....
00000000.
000000...
0000.....
```


A side benefit of this tab/space conversion is the ability to convert all the Tab characters in the edit buffer into spaces, or convert sequence of spaces into Tab characters. First mark the entire edit buffer as a columnar block; then move the block into a Text Register. If you only want spaces in the file, disable

{CONFIG, Emulation, Retab after columnar block insert} and insert the Text Register back in. If you want spaces converted to Tab characters, enable

{CONFIG, Emulation, Retab after columnar block insert} and insert the Text Register.

Search and Replace

The descriptions for **{SEARCH, Search}** and **{SEARCH, Replace}** in Chapter 5 (Menu Reference) describe in detail how to perform normal search and replace operations. Therefore, this discussion assumes you already know the basics of “search strings” and “replacement strings”.

This topic describes the powerful search and replace modes in VEDIT called *regular expression pattern matching* and *short form pattern matching*.

When the configuration parameter **{SEARCH, Config, Regular expression search mode}** is set to “Yes”, (the default) VEDIT is in *regular expression* pattern-matching mode. When this is set to “No”, VEDIT is in *short form* pattern-matching mode. All search and replace operations take place in one of these two modes.

Regular Expressions

Regular expressions are a type of text pattern matching originally developed within the UNIX environment. They can be used as an alternative to (but not in conjunction with) VEDIT’s *short form* pattern matching. The UNIX specifications are followed quite closely.

In QNX VEDIT, *regular expression* pattern-matching is the default.

The choice between short form pattern matching and regular expressions is partly a matter of personal preference and a consideration of the advantages and disadvantages of each.

Advantages of Regular Expressions:

- For the most part, more flexible and powerful than normal pattern matching. Particularly powerful are the constructs `+` meaning “one or more occurrences of” and `*` meaning “zero or more occurrences of”.
- During a search and replace, groups of characters matched during the search can be used as part of the replacement text in very flexible ways. *This is probably the biggest advantage of regular expressions.*
- A consistent or “regular” syntax, which, with a little experience, makes them easy to understand. Short form pattern matching codes are cryptic.
- Compatibility. Other QNX tools which use regular expressions include the **grep** text search utility, **awk** text processing language, **sed** batch editor and the **lex** compiler development tool. Many editors and other tools on other systems also use them.

Disadvantages of Regular Expressions:

- The pattern being searched must all be on one text line; cannot search for multi-line patterns.
- For the most part, regular expression searching is slower than short form pattern matching. This may be a factor worth considering in complex macros.
- Regular expressions can be lengthy and verbose. For example the short form pattern “`|A`” is equivalent to the regular expression “`[a-zA-Z]`”.
- Many characters have a special meaning in regular expressions. A special syntax must be used when searching for these characters.

Regular Expression Basics

Several characters have special meaning in regular expressions:

- | | | |
|----|----------|---|
| . | (period) | Simple wildcard that matches any character. |
| ^ | (caret) | Matches the beginning of a line (when it is the first character in a regular expression). |
| \$ | | Matches the end of a line (when it is the last character in a regular expression). |
| * | | Matches zero or more occurrences of the <i>preceding</i> character (or list). |
| + | | Matches one or more occurrences of the <i>preceding</i> character (or list). |
| ? | | Matches zero or one occurrences of the <i>preceding</i> character (or list). |

Examples:

- | | |
|--------------|---|
| o.e | Matches any text containing “o” followed by <i>any character</i> followed by “e”. Will match the word “one” and the “ome” in the word “some”. |
| ^o.e | Matches the same strings, but only if they appear at the beginning of a line. Will match the word “one” at the beginning of a line. DOES NOT match “some”. |
| o.e\$ | Matches the same strings, but only if they appear at the end of a line. Will match the word “one” and the “ome” in the word “some” if they occur at the end of a line. DOES NOT match the word “something”. |
| ^\$ | Matches only a blank line. |
| an*d | Matches any text containing “a” followed by <i>zero or more</i> occurrences of “n” followed by a “d”. Will match the “ad” in the word “add” and the word “and”. |

an+e	Matches any text containing “a” followed by <i>one or more</i> occurrences of “n” followed by an “e”. Will match the “ane” in the word “cane” and the “anne” in the word “banned”.
an?e	Matches any text containing “a” followed by <i>zero or one</i> occurrences of “n” followed by an “e”. Will match the “ae” in the word “Caesar”, the “ane” in the word “cane”, but NOT the word “banned”.

You can think of “*” as indicating that one or more occurrences of the previous character are “optional”. Notice that “**ann*e**” is identical to “**an+e**”.

A list of characters surrounded by the square brackets “[” and “]” matches any *one* character in that list. Within square brackets, a range of characters may be abbreviated using a hyphen “-”. However, when the first character in the list is a “^” (caret) the list matches any character *except* those in the list.

[abc]d	Matches “ad”, “bd” or “cd”.
[a-z]	Matches any lowercase letter.
[A-Za-z]	Matches any letter, uppercase or lowercase.
[A-Z][a-z]*	Matches all capitalized words. It matches an upper case letter followed by zero or more lower case letters.
[^a]	Matches any character <i>except</i> “a”.
[^a-z]	Matches any character <i>except</i> a lower case letter.
[^0-9]	Matches any character <i>except</i> a digit.
\[	Matches the “[” character.
\\	Matches the “\” character.

A “\” followed by any character (except a digit or letter) simply matches that character. This allows searching for those characters which are used as special symbols in regular expressions.

IMPORTANT NOTES

These characters are special symbols in regular expressions and therefore must be preceded by “\” in order to search for them in the text:

`^ $ . * + ? - ~ \ | [ ] { }`

The syntax allows exceptions where the “\” is not needed, but in those cases the “\” does not hurt either, and we recommend using it.

On the replacement side of a search and replace command, only the two characters “\” and “&” are special symbols.

The special character “|” is the “OR” operator which may occur between two sub-expressions. The entire expression then matches any text that is matched by the preceding sub-expression *or* the following sub-expression. Note: “|” cannot occur within “{ }”; each sub-expression must by itself be a valid expression.

man | woman Matches the word “man” *OR* the word “woman”.

a+ | b+ Matches one or more occurrences of “a” *or* of “b”.

Groups and Replacement Strings

The special symbols “{” and “}” group regular expressions for reference purposes. They permit the text matched by the expression within “{ }” to be referenced again in the search string or to be included as part of the replacement text.

An expression may contain up to 9 groups which are referenced by number using the constructs “\1” to “\9”. The groups are numbered in the order of their opening braces.

Groups may be referenced in a latter part of the regular expression; this allows the notation to handle repetition of parts of a pattern. They may also be used in the replacement string. This allows portions of the matched text to be used as parts of the replacement text. Groups may also be nested.

The character “&” has a special meaning only in a replacement string and references the *entire text* matched by the search.

Consider the expression “**the {man} | the {woman}**”. If the matched text is “the man”, “\1” is “man” and “\2” is empty (null).

Now consider the expression “**{the { [a-z] + } } has**”. If the matched text is “the woman has”, “\1” is “the woman” and “\2” is “woman”.

<code>{ [a-z] [a-z] [a-z] }\1</code>	Matches two contiguous occurrences of the same three letters. Will match “nownow”, “powpow”, etc.
<code>{ . * }\1</code>	Matches any repeating text. Will match “nn”, “nownow”, “12341234”, etc.
<code>^ { . * } \1 \$</code>	Matches a line consisting of two repeated occurrences of the same text (identical right and left halves).

Complete Examples

These examples show search and replacement strings using regular expressions. They illustrate how groups of matched text can be used as part of the replacement string.

Find? `[A-Z] [a-z] *`

Search for the next capitalized word. Note that it will also match the single letter word “A”.

Find? `0x[0-9a-fA-F] +`

Search for the next hexadecimal number in a “C” program. It searches for “0x” followed by one or more hexadecimal digits.

Find? `.*,`

Replace with? (None, just press <Enter>)

Delete all text up to a comma on the next line which contains a comma. (Remember that the entire matching text must occur on one line.)

Find? `{[Hh]}ello`

Replace with? `\li`

Searches for “Hello” or “hello” and replaces it with “Hi” or “hi” respectively.

Find? `[Hh]ello`

Replace with? `&~world`

Searches for “Hello” or “hello” and replaces it with “Hello~world” or “hello~world” respectively. The “&” references the entire text matched by the search. Note that the grouping characters “{ }” are NOT needed in order to use “&”.

Regular Expressions -
More Details

The following special matching characters are defined:

<code>\b</code>	Matches the ASCII backspace character (value 08).
<code>\dDDD</code>	Matches the character with decimal value 'DDD'. All three digits <i>must</i> be present.
<code>\e</code>	Matches the ASCII ESC character (value 27).
<code>\hHH</code>	Matches the character with hexadecimal value 'HH'. Both digits <i>must</i> be present.
<code>\r</code>	Matches the ASCII Carriage-Return character (value 13) which is not part of the CR-LF newline sequence.
<code>\s</code>	Matches the ASCII space character (value 32).
<code>\t</code>	Matches the ASCII Tab character (value 09).

<code>\0</code>	Matches the ASCII null character (value 00).
<code>[z-a]</code>	A range may also be specified in reverse order.
<code>[~a-z]</code>	Matches any character except for a lower case letter—“ <code>[~]</code> ” is equivalent to “ <code>[^]</code> ”.

Although we recommend using the “`\`” in front of any special symbols you need to search in the text, the “`\`” is not needed when there is no possibility of confusion. For example, the characters “`{`”, “`}`”, “`|`”, “`*`” and “`+`” are not special within the square brackets. The “`$`” is only special at the very end of the expression. Even the hyphen “`-`” is not special immediately following “`[`” or preceding “`]`” or outside of square brackets.

The configuration switch

{SEARCH, Config, Equate Upper/Lower case during search} is applicable to regular expressions, but not within bracketed lists. Therefore, “**hi**” will match “**HI**”, “**Hi**”, etc., and the expression “**[a-z]i**” will match “**hi**” and “**hI**”, but not “**Hi**”. There is little reason to ever turn the switch off—you could use the expression “**[h][i]**” to search for the lower case word “**hi**”.

Operator Precedence

The order of precedence of the operators is:

<code>\</code>	Highest
<code>[]</code>	
<code>* + ?</code>	
<code>{ }</code>	
Concatenation	
<code> </code>	Lowest

One *important rule* is that regular expressions *cannot* match **newline** sequences at the end of a line. Therefore, the expression “**[^a]**” matches any character except for “**a**” *and* a **newline** (single Line-feed or Carriage-return and Line-feed).

<code>[\h00-\h1f]</code>	Matches any control characters except newline .
--------------------------	--

The *most important rule* about regular expressions is that the “*” and “+” operators always match the longest possible string that still allows the rest of the expression to match. Consider the rather subtle expression (already used in an example above):

```
{.*}\1
```

and the text line:

```
a00a abc12231223cba
```

On the first search, it will match the “00” in “a00a” since it matches any repeating text. However, on the second search it does not match the “22”, but rather the entire “12231223” because this is the longer repeating text string that includes “22”.

(This rule has tremendous implications and unfortunately slows down searches significantly—the search cannot stop on the first match, but must rather keep looking for ever longer matches. It is probably because of this rule that the original UNIX designers decided to restrict searches to a single text line—a multi-line search for “{.*}\1” in even a 1K file can involve over a million comparisons. Many other implementations of regular expressions on micro-computers do not closely follow this rule.)

Short Form Pattern Matching

VEDIT’s *short form* pattern matching makes it possible to search not only for particular characters, but also for types of characters such as “any digit”, or for characters that meet special conditions such as “occurring at the beginning of a line”. You could even search for any five letter word beginning in “t” and ending in “n”.

These generalized searches are performed by using “*pattern matching codes*” within the search string. Each pattern matching code consists of the special character “[” followed by another character—typically a mnemonic letter.

NOTE:



Although the mnemonic letter may be entered in upper or lower case, for purposes of clarity, all examples show these letters in upper case.

Here are a few examples of search strings using pattern matching:

D D	Search for two consecutive digits.
N D D D N D	Search for next two digit number. (It will not match a three digit number.)
<note	Search for the word “note” appearing at the beginning of a line.
t A A An	Search for any five letter word beginning in “t” and ending in “n”.

The complete set of Pattern Matching Codes are:

A	Match any alphabetic letter, upper or lower case.
B	Match a blank - a single Space or Tab.
C	Match any Control Character - a character with an ASCII decimal value of 0 to 31.
D	Match any numeric digit—“0” through “9”. This code does not match “.” or “,”.
F	Match any alphanumeric character - a letter or a digit.
G	Match any group character - { }, [], <> and (). (Internally used by {GOTO, Matching ()}.)
L	Match the newline character. When {CONFIG, File Handling, Newline character(s)} is set to “0” (DOS files), this matches a Carriage-Return, Line-Feed pair as a single newline character. Otherwise it matches a single Line-Feed character.
M	Match multiple characters - zero, one or more characters until the string following the “ M” is satisfied.

N	Match any character except the following character or pattern code. Use this code to exclude a certain character or type of character. For example, to search for “exam “ or “examiner” but not “exams”, use “exam Ns”. Think of “ Nx” as “not x”.
Rr	Use the contents of Text Register ‘r’ in this position in the search string.
S	Match any separator - a character which is not a letter or a digit.
T	Match selected separators (terminators). This is similar to “ S” but will only match the following characters: ; : , ' " Space Tab Carriage-return Line-feed This code is primarily useful for programming language constructs where characters such as “\$” and “_” may be part of variable names and labels.
U	Match any upper case letter.
V	Match any lower case letter.
W	Match “white space”—one or more Spaces and/or Tabs.
X	Match any single character; this is the simple “wildcard” similar to “?” in QNX filenames.
Y	Match zero, one or more characters until the immediately following character or pattern matching code is satisfied.
0	Match the “Null” character (ASCII 0).
{set}	Match any one item in the “pattern set”.
[set]	Match one optional occurrence of any item in the “pattern set”.
	Match the “ ” character. Since “ ” starts a pattern matching code, you need two together to search for a single “ ” in your text.

Most pattern matching codes match a single character; however, some will match zero, one or even many characters. The code “|W” matches “*white space*”—one or more spaces and/or Tab characters. For example, search and replacement strings to replace all white space at the beginning of a line with a single Tab character are:

Find? |<|W

Replace with? <tab>

The codes “|<” and “|>” match zero characters; they only ensure that the entire search string matches at the beginning or end of the line, respectively.

Remember when using the **REPLACE** function that *all* text characters that are matched by the search string are replaced by the new text. This is where the difference between “|<” and “|L” is important—text matched with “xxxx|<” does not include the **newline** character, while text matched with “xxxx|L” does include the **newline**.

The code “|M” is useful for finding text where the beginning and end are defined, but the middle does not matter. For example, let's say you want to check that all quote marks are properly paired. Press **FIND** and enter the following search string:

"|M" (That's four characters)

The cursor should be at the end of the first quotation. Each time you press **FIND AGAIN** the cursor should move to the end of the next quotation. If it doesn't, a quote mark is not properly paired.

Besides being useful in searches, the “|M” code can be used to delete large blocks of text. For example, the following search and replace string would delete this paragraph:

Find? Besides|Mparagraph:

Replace with? (None, just press <Enter>)

Often the “|**M**” code is too unrestricted. For example, the search string “|**Sa|Mtion|S**” will match words beginning in “a” and ending in “tion”. However, it will also match the next word beginning in “a” followed by any text until it finds a word ending in “tion”.

The code “|**Y**” can also match multiple characters, but is more restrictive than “|**M**”. “|**M**” matches ever more characters until the rest of the search string is satisfied, or the end of the file is reached. Once that portion of the search string in front of the “|**M**” is matched, it is never searched for again; there is no need. On the other hand, “|**Y**” matches ever more characters *only until the very next character (or pattern) matches*. If the rest of the search string then fails, the entire search string is re-searched.

For example, in assembly language programming, any text following a “;” character is a comment. Instructions are often followed by a few tabs (to align the comments), the “;” and the comment. The following search and replacement strings will delete the tabs (and/or spaces) and the comment which follow any instruction. However, lines which are entirely comments (i.e. that have a “;” in the first column) are not deleted.

Find? |**W**;|**Y**|

Replace with? (None, just press <Enter>)

As another example, we want to search for the following two lines:

MOV BL,DL ;An arbitrary comment

MOV BH,DH

We want to be certain that the second line immediately follows the first line. As indicated, the critical part of the first line could be followed by unknown text. The search string to find these two lines is:

MOV BL,DL|Y|MOV BH,DH

Notice that substituting |**M** for |**Y** would not perform the same function—we could no longer be sure that the second line immediately followed the first line.

Pattern Sets

The pattern matching codes `{set}` and `|[set]` contain a user-definable *pattern set*. This is analogous to an **OR** operator—the match is successful if the text matches the first item *or* the second item **OR** the third item, etc. Each item in the pattern set can itself be a search string. The items are separated from each other by commas `,`. Commas themselves are represented by `|,`.

For example, the search string to find any of the animal names “CAT”, “DOG”, “LION” or “MOUSE” is:

```
| {CAT, DOG, LION, MOUSE}
```

Pattern sets are very useful when searching for alternative words. Unfortunately, pattern sets execute more slowly than other searches—in this example the pattern set has to be checked for each character in the text.

As another example, we want to search for occurrences of the words “automation” and “automobile”. We could of course place the entire words into a pattern set. As an alternative, we will just place the word endings into a pattern set in the following search string:

```
auto|{mation, mobile}
```

Since the pattern set is only checked when “auto” has already been found, this search will run very quickly.

The pattern matching code `|[set]` matches one optional occurrence (i.e zero or one occurrence) of any item in the pattern set. For example, the search string

```
the |[tall, short, fat, thin] man
```

would match “the man”, “the tall man”, “the short man”, “the fat man” and “the thin man”. However, the string

```
the |{tall, short, fat, thin} man
```

would not match just “the man”.

Note that neither string matches “the short fat man”.

For yet more complex types of searches you may want to use “Regular Expressions”.

Using Text Registers in Search Strings

The contents of a Text Register or edit buffer can be used as part of a search string. The register contents are accessed with the pattern matching code “|**R***r*” where ‘*r*’ is the name of the register to use. This makes it possible to have “variable” search strings.

For example, assume the following Text Register contents:

Register 1 contains: “**Nice**”

Register 2 contains: “**a walk**”

Then the following three search strings are all equivalent:

Nice night for a walk

|**R1** night for a walk

|**R1** night for |**R2**

Word Processing Functions

In addition to its wide range of features for general purpose editing, VEDIT has features specifically designed to assist with word processing. These include:

Left Margin for Indented Text

When a left margin is set, VEDIT indents new lines of text by automatically adding tabs and spaces (or just spaces) to reach the left margin. Existing paragraphs and arbitrary blocks of text can also be indented.

Right Margin and Word Wrap

Setting a right margin also enables “word wrap”—words which would exceed the right margin are instead “*wrapped*” to the next line without breaking the word in half. VEDIT performs word wrap by inserting a normal **newline** character before the wrapped word—just as if you had pressed <Enter>.

Paragraph
Formatting

If desired, each paragraph can have different left and right margins. The margins for an existing paragraph can also be changed and the paragraph then “re-formatted” to fit the new margins.

Justification

When paragraphs are formatted, they can optionally also be justified. Justification produces a straight right edge to the text by adding spaces between words on each line.

Center Line

Lines of text can quickly be centered between the current left and right margins with the {EDIT, Center Line} function.

Unlike dedicated word processors, VEDIT’s word processing functions never insert special “control codes”. Paragraphs are formatted by inserting only **newline** characters. Text is indented by using only Tab characters and spaces; if desired only spaces can be used.

This makes it possible to use text files formatted with VEDIT with almost any other program. For example, VEDIT is ideal for preparing and editing text files using with Desktop Publishing packages such as Xerox Ventura Publisher(tm).

**Definition of
“Word” and
“Paragraph”**

It is important to know exactly what VEDIT considers to be a word and a paragraph.

Words are allowed to have embedded periods in them, as in “i.e.”. A comma “,” always ends a word, even if the comma is not followed by a space. As a special case, numbers with embedded commas, such as “10,000” are treated as one word. Of course spaces, tabs (“white space”) and “newlines” separate words from each other. All other characters are allowed in words.

To VEDIT, a paragraph is separated from preceding or following paragraphs by at least one blank line or a line which starts with a “.” “!” or “@” in the first column. Lines starting with these characters are assumed to be “print formatting commands” for programs like **nroff** or **Ventura Publisher**. Print formatter command lines are not considered part of any paragraph and are therefore never re-formatted.

Indenting Text (Left Margin)

There are several ways to indent text so that it doesn't begin in the first column. You can, of course, type spaces at the beginning of each line to be indented. This is the normal way of indenting the first line of a new paragraph. However, VEDIT can indent each new line of text for you, automatically. This is useful in word processing for indenting entire paragraphs, and for editing programs written in structured languages such as C.

The left margin determines how much newly entered lines will be indented. Normally the left margin is set to column 1. Pressing **INDENT** (<F8>) increases the left margin and **UNDENT** (<F7>) reduces the left margin. These functions change the left margin by the “*indent increment*”, which is normally “4”, but configurable.

The left margin can also be changed with
{CONFIG, Word Processing, Left margin}.

VEDIT indents text by padding the beginning of lines with Tab characters and spaces. The optimum number of tabs and spaces will be used and depends upon the currently set tab stops. If you prefer text to be indented using only spaces, set
{CONFIG, Emulation, Expand Tab key with spaces} to “Yes”.

VEDIT has another mode of indentation called “*Auto-Indent*”. It is primarily intended for editing structured programming languages such as “C”. In Auto-Indent mode each new line will be indented the same amount as the previous text line. You can then change the indentation of the new line by pressing **INDENT** or **UNDENT** to start or end a block of program instructions. The main advantage of Auto-Indent mode is that you can jump around in a program and newly entered instructions will automatically fit the indentation of the current block of instructions.

Auto-indent and the “Indent increment” are configurable with {CONFIG, Programming, Auto-Indent mode} and {CONFIG, Programming, Indent increment}.

HINTS:



Do not indent your text to keep it from printing on the left edge of the paper. The print functions let you set a separate *printer margin* for this purpose.



INDENT is not designed for indenting the first line of a paragraph. Spaces or tabs should be used for that purpose.

Word Wrap (Right Margin)

Setting the right margin enables “*word wrap*” which automatically starts a new line when text reaches the right margin. The entire word, which would otherwise exceed the right margin, is wrapped to the next line without breaking the word in half. The new line will start at the current left margin.

Ordinarily, the right margin is disabled. To enable it, use {CONFIG, Word Processing, Right margin}.

The right margin is the last column in which a displayable character can occur. However, spaces, especially the typical two spaces following a sentence, are allowed to exceed this margin.

A value of zero turns the right margin off, disabling both word wrap and the {EDIT, Format Paragraph} function.

The right margin can be greater than the screen line length, in which case VEDIT will either scroll the screen horizontally or display a continuation line before the word wrap takes place.

As you edit a paragraph, existing text will sometimes extend past the right margin. This is normal, because word wrap only occurs when new text is entered past the right margin. You can get all the text back between the margins by reformatting the paragraph.

IMPORTANT NOTES:

The right margin should be turned off when editing programs; otherwise, accidentally selecting **{EDIT, Format Paragraph}** will generally scramble your program. VEDIT's "undo" usually cannot recover the text in this case.

Don't confuse the right margin with the "Horizontal Scroll Margin"—the latter only controls how long lines are displayed on the screen.

Formatting and Justifying Paragraphs

VEDIT can "format" a paragraph to fit it between the current left and right margins. To do this, VEDIT may move one or more words from the end of a line to the beginning of the next line or borrow words from the beginning of the following line to fill out a line. In the process, the number of lines in the paragraph may change. VEDIT can optionally add extra spaces between words to "justify" (straighten) the right edge of the paragraph.

The paragraph will be formatted to fit between the current left and right margins. For example, if a paragraph is indented 8 spaces before formatting and the left margin is currently 4, formatting the paragraph will result in it being indented only 4 characters.

To format a paragraph, first ensure that the left and right margins are set as desired, place the cursor anywhere in the paragraph and select **{EDIT, Format Paragraph}** (<Ctrl>-<F>). After formatting, the cursor is advanced to the beginning of the next paragraph so that you can easily format several consecutive paragraphs.

After formatting a paragraph or using word wrap, you may notice that each text line ends in a space. This actually is the space you typed between the words. Lines ending in a period "." are typically followed by two spaces. These spaces at the end of a line are allowed to exceed the right margin.

When formatting paragraphs, VEDIT attempts to maintain double spaces between sentences, i.e. after ".", "!" and "?".

NOTE:



An indented line is not enough to separate one paragraph from another; blank lines *must* be used to separate paragraphs. Otherwise, re-formatting paragraphs will combine several paragraphs into one!

Offset Paragraphs

An “Offset Paragraph” is a special type of paragraph whose first line is not indented the same distance as the rest of the paragraph. VEDIT maintains this difference in indentation when formatting a paragraph as in the following examples.

This is an offset paragraph. Notice how the first line is indented 4 character spaces further than the rest of the paragraph.

When this paragraph is reformatted with different margins, the second line will be aligned with the left margin and the first line will be indented the same four spaces.

An offset paragraph can also have the first line outdented from the remainder of the paragraph.

1. This is an offset paragraph. Notice how the first line is at the left margin while the remaining lines are indented 4 characters.

When this paragraph is reformatted with different margins, the first line will be aligned with the left margin.

When working with numbered offset paragraphs (as in the example above), you may find it helpful to place non-space characters between the number and the paragraph body. This ensures that first word of the paragraph is not misaligned with rest of the paragraph body when using justification.

Justification

When “*justification*” is enabled with {CONFIG, Word Processing, Justify paragraphs}, VEDIT will adjust the spacing between words to create a straight right edge to the paragraph.

This is a justified paragraph. Notice how the spacing is randomly adjusted between words to maintain a straight right margin. This is preferable in some, but not all applications.

If you need to edit text after it has been justified, it is easier if you first “*unjustify*” the text. This removes the additional spaces between words, leaving the right margin ragged. (It will leave two spaces following “.”, “?” “:” and “!”). To unjustify a paragraph, change {CONFIG, Word Processing, Justify Paragraphs} to “2” and reformat the paragraph.



Documents justified by another word processor should first be unjustified for easier editing with VEDIT.

Documents which will later be formatted by another program such as a Desktop Publishing package should generally *not* be justified with VEDIT.

Editing Multiple Files

You can simultaneously edit two or more files, displaying just one at a time using the entire screen, or several at a time in separate windows.

The following Keystroke Macros are built into the default keyboard layout to simply the editing of multiple files. This topic and the following “Windows” topic describe the use of these Keystroke Macros in more detail.

- | | |
|------------------------------|--|
| <Alt>-<N> | Edit a new file in the current buffer. You will be prompted whether the current file is to be saved or abandoned. Use this when you are done editing the current file and want to edit another file. Same as {FILE, New File}. |
| <Alt>-<O> | Simultaneously edit another file. It will automatically choose the next available buffer. Same as {FILE, Open File}-N. |

<Alt>-<Y> Same as <Alt>-<O> except that it also splits the screen horizontally into two windows.

Same as **{FILE, Open File}-Y-H**.

<Alt>-<T> Toggles between the files being simultaneously edited.

Editing Multiple Files Without Windows

The initial file you edit is loaded into the “*main edit buffer*” (buffer @). The contents of the edit buffer are what you see on the screen. In addition to the main edit buffer, you can also use any of the 36 Text Registers as additional “edit buffers” for simultaneously editing multiple files.

Any Text Register can be converted for use as an edit buffer; however by convention, only “A” through “Y” should be used.

The main difference between Text Registers and edit buffers is that edit buffers can be used to edit files and Text Registers can’t. Other differences are explained under “Edit Buffer and Text Register Differences” in this chapter.

Editing multiple files is analogous to having VEDIT running on several computers at once—switching edit buffers moves you from one computer to another. Switching to an edit buffer makes it the “*current*” or “*active*” edit buffer. Only one edit buffer is active at a time—in other words, editing changes only take place to one file at a time. When an edit buffer other than the main edit buffer “@” is active, its name is displayed on the left side of the status line.

To simultaneously edit another file:

1. Select **{FILE, Open File}**. Since we don’t want windows, press “N” after the “Open new window too (Y/N)” prompt.

OR

1. With the default keyboard layout, press <Alt>-<O>.

Note that this built-in Keystroke Macro is not listed as a “hot key” in the **{FILE}** menu because it does a little more—it accesses the “Open File” item and also answers the next prompt with “N”.

2. At the “Enter Filename” prompt, enter the name of the file to be edited, or press <Enter> to select a file using “Point and Shoot” file selection.

VEDIT automatically selects the next available Text Register and converts it for use as an edit buffer for editing the selected file. Beginning with “A”, VEDIT selects the next Text Register which is not already an edit buffer and is empty.

OR

1. Experienced users can select **{FILE, Buffer Switch}** (<F4>) and, when prompted, type the name of either an existing edit buffer or the name of a Text Register to be converted for use as an edit buffer. This makes the selected edit buffer “active” or “current”.
2. Select **{FILE, New File}** (<Alt>-<N>). If a file is already being edited in the current edit buffer, you are prompted whether to save or abandon the file before loading the new file. You are then prompted for the file name.

It is usually easier to have VEDIT automatically select the next edit buffer for you; the particular names of the edit buffers is rarely important.

In either case, the name of the new file is now displayed on the status line. The name of the newly “active” edit buffer is displayed on the right side of the status line following the “#”.

You can repeat the above procedure until all desired files are loaded. To obtain a list of all edit buffers and associated file names, use **{HELP, Edit Buffers}**.

Switching Between Files (Edit Buffers)

To switch to another file:

1. With the default keyboard layout, press <Alt>-<T>. This toggles you, round-robin fashion, to the next edit buffer and the file (if any) being edited in it.

OR

1. Select **{FILE, Buffer Switch}** (<F4>); at the prompt "Enter Edit Buffer Name" type the single character name of the edit buffer containing the desired file. You can easily select the main "@" edit buffer by pressing <Enter> instead of the edit buffer name.

Unless you remember which file is in which edit buffer, it is easiest to use the built in <Alt>-<T> keystroke macro in the default keyboard layout. (Other keyboard layouts have a corresponding key.)

Exiting and Quitting Multiple Files

To exit VEDIT and save all files being edited:

1. Select **{FILE, Exit}**. At the prompt, type "A" to select "All files". This will save all files and exit back to whatever program called VEDIT, typically the QNX shell.

Alternatively, you can select the equivalent **{ESCAPE, Exit All Files}**.

To exit VEDIT and quit (abandon) all files being edited:

1. Select **{FILE, Quit}**. At the prompt, type "A" to select "All files". You will be prompted for confirmation; type "Y" if you really want to quit and abandon any changes you have made to the files being edited.

You can also exit or quit your files one at a time with **{FILE, Exit}** and **{FILE, Quit}** by typing "Y" instead of "A" at the prompt.

If you exit or quit the current edit buffer while other edit buffers are in use, you are automatically switched to one of the remaining edit buffers. When no other edit buffers remain, these functions exit VEDIT.

Consider the following example. You are editing two files, one in the main edit buffer “@” and one in buffer “A”. You are currently in buffer “A” and select {**FILE, Exit**}.

This will save the file in buffer “A” and switch you to the main buffer “@”. You again select {**FILE, Exit**} to save the second file. Since you are exiting the main edit buffer and no other edit buffers are in use, you will be returned to the program which called VEDIT—typically the QNX shell.

Exiting or quitting from an edit buffer converts the edit buffer back into an empty Text Register. Only the main edit buffer will not convert into a Text Register.

Simultaneous Editing with Windows

You can have each file displayed in its own window. Of course, this reduces the amount of a file that can be displayed at any one time. However, you can “Zoom” a window to fill the full screen when you need to see more.

To edit an additional file in its own window:

1. Select {**FILE, Open File**}. Since we want windows, press “Y” after the “Open new window too (Y/N)” prompt.

Type “H” to have the screen (or current window) split into two horizontal (top/bottom) windows. Type “V” for vertical (side by side) windows.

OR

1. With the default keyboard layout, press <Alt>-<Y>. Note that this built-in Keystroke Macro is not listed as a “hot key” in the {**FILE**} menu because it does a little more—it accesses the “Open File” item and selects horizontal windows.
2. At the “Enter Filename” prompt, enter the name of the file to be edited, or press <Enter> to select a file using “Point and Shoot” file selection.

The current window will have been split into two windows with the new file displayed in a new window.

Each window's border displays the name of the window and the name of the file. In this case, the name of the window is the same as the name of the "attached" edit buffer. The "active" window, in which you are currently editing, has the border word "WINDOW" highlighted.

You can also create custom sized windows with the {WINDOW, Create} function described in Chapter 5 (Menu Reference).

You can switch between files in the same way as before or you can use a mouse.

To switch between windows with a mouse:

1. Move the mouse anywhere within the desired window and click the left button.

This switches you to the file displayed in the selected window.

Copying Text From One File to Another

A common reason for editing two files is to copy portions from an existing file to a new file. This requires the use of a Text Register.

Copying text from one file to another:

1. Using the technique described for {BLOCK, Copy to Register}, copy the block to a Text Register (suggest "0" through "9").
2. Switch to the second file and move the cursor to the desired position.
3. Select {BLOCK, Insert Register} or press [T-REG INSERT] (<F10>).

Edit Buffer and Text Register Differences

The function {FILE, Buffer Switch} switches to the selected Text Register and converts it into an edit buffer if it is not already. A Text Register remains an edit buffer until you explicitly exit/quit with {FILE, Exit} or {FILE, Quit}.

There are several reasons for converting Text Registers into edit buffers:

- The primary purpose is for editing additional files.
- Converting a Text Register into an edit buffer preserves its contents. Therefore, you can copy a block of text into a Text Register, edit the text, and then insert it into any desired file(s).
- Files containing “Command Macros” (macros written in the VEDIT macro language) can be loaded into the edit buffer and then executed.

Many Text Register operations can also be performed on edit buffers—you can insert the contents of an edit buffer with **{BLOCK, Insert Register}** or execute it as a Command Macro with **{MISC, Execute Macro}**.

*The main difference is that you cannot change the contents of an edit buffer except when it is the “active” edit buffer. Therefore, you cannot copy a block of text into an edit buffer with **{BLOCK, Copy to Register}**.*

This limitation helps prevent you from inadvertently altering the corresponding file.

Windows

Windows are sections on the screen which are primarily used for displaying the file(s) being edited. Windows can either simultaneously display multiple files and/or different portions of one file. Each window has a single character name. Excepting multiple windows per file, windows usually have the same name as their corresponding edit buffer.

When simultaneously editing multiple files, **{FILE, Open File}** gives you the option of splitting the screen into windows. However, the **{WINDOW}** menu gives you additional flexibility to create windows of the exact size you want. For example, you might want to have a

large window at the top of the screen for the main file you are editing and a small window at the bottom for a reference file or additional notes.

New windows are created by splitting an existing window into two, either vertically or horizontally. Any resulting windows may then be split further and may be as small as one line and/or 10 columns. Minimal borders are drawn on the top and left to separate windows from each other. Overlapping windows are not supported.

The following example gives the steps necessary to create a custom sized window and edit a file in it. It assumes you are currently editing only one file (NEWS.TXT).

To edit an additional file in a custom window:

1. Select **{WINDOW, Create}**.
2. Type **"R"** to place the new window at the right side of the current window and type **"A"** for the window name. Enter **"30"** for the number of columns.

You should now have an empty 30 column window at the right of the screen.

3. To edit a file in this window select **{FILE, Buffer Switch}** (<F4>). At the prompt "Enter Edit Buffer Name:" type **"A"**.

This switches to edit buffer **"A"** and to the new window which is still empty.

4. Select **{FILE, New File}** and enter the desired file name.

The new file should now be displayed in the new window.

To display an edit buffer in its own window, give the window the same name as the edit buffer. Otherwise the edit buffer will display in the current window. In effect, edit buffers "attach" to windows of the same name.

Windows and Edit Buffers

Windows and edit buffers are associated with each other. When an edit buffer (file) is displayed, a window in which to display the file is selected according to these ordered rules:

1. If the edit buffer is already “attached” to a window, and the window still exists, it selects this window. In the case that the edit buffer is attached to multiple windows, it select the “primary” window.
2. If a window by the same name as the edit buffer exists (and is not already attached to another edit buffer), it selects and attaches itself to this window. According to rule #1, it will continue to use this window as long as possible.

For example, if you create the window **A**, then the next time you switch to edit buffer **A**, it will be displayed in window **A**.

3. It temporarily selects the current window.

Therefore, an edit buffer tends to “attach” itself to a window by the same name. If this preferred window does not exist, it will temporarily use the current window. This selection is temporary; the edit buffer continuously checks for a recently created window by the same name.

HINT:



To keep things simple, it is best to always create windows with the same name as the edit buffers to be displayed in them. We also recommend using **A** through **Z** for edit buffers. Reserve **0** through **9** for simple Text Registers.

Deleting Windows

{WINDOW, Delete} deletes a window without affecting the corresponding edit buffer—you can continue editing the file that was displayed in the deleted window. The edit buffer that was displayed in the window will now share a window with another edit buffer.

{WINDOW, Remove All} deletes all windows (except window “@”) and initializes the screen.

The “main” window (@) cannot be deleted. When VEDIT starts up you are in the main @ window; since it covers the entire screen no window border is displayed.

Zooming A Window

After you have created several windows, it is often helpful to “zoom” the current window to fill the entire screen for easier editing.

To zoom the current window to full screen:

Select **{WINDOW, Zoom}**.

The zooming remains in effect until you switch to another window or select **{WINDOW, Zoom}** again to “un-zoom” the screen to redisplay all windows.

Editing One File in Two Windows

VEDIT can display a file in more than one window at a time. This lets you view different sections of a file at the same time. For example, you may want to refer to the definitions listed at the beginning of a file while you are editing in the middle of the file.

- **{WINDOW, Horiz. Split}** creates a new window in the lower half of the current window for displaying the same file.
- **{WINDOW, Vert. Split}** (<Alt>-<V>) creates a new window in the right half of the current window for displaying the same file.

VEDIT automatically assigns each such window a unique name—“0”, then “1” and so on. Use **{WINDOW, Switch}** (<Alt>-<F4>) to switch between these windows.

This process may be repeated to display additional parts of a file.

All windows displaying a common file update dynamically when the text in their displayed region changes. Each window also displays its own cursor and can be scrolled independently of the other window(s). Notice that only the cursor in the active window moves. The cursor(s) in the inactive window(s) indicate your editing position when you switch to those windows.

These windows can be deleted using **{WINDOW, Delete}**. This does not affect the contents of the file.



Chapter 4

Command Macros

Introduction

Command Macros are sequences of commands written in the VEDIT macro language.

This chapter covers everything the VEDIT user needs to know about Command Macros—primarily how to load and run the supplied macros.

This chapter assumes you understand the fundamentals of starting VEDIT and Text Registers, both covered in Chapter 3 (Editing Guide).

“Ready to use Macros” in Chapter 1 (Introduction) gave a short overview of the Command Macros described in this chapter.

VEDIT Macro Language

The VEDIT macro language is a complete text-oriented programming language. It has arithmetic capabilities, testing, if-then-else decision making, looping, user input, screen output, window control and much more. Its over 100 commands can perform almost any conceivable character, line, block or file operation.

The main editing mode of VEDIT is called *Visual Mode*; VEDIT derived its name from “Visual EDITor”. In this mode most keys you hit are characters to be inserted in the buffer. Only the “hot keys” with “keystroke macros” attached to them execute macro language commands.

You can directly access the macro language at any time by pressing a key (default <Alt>-<F8>) to enter *Command Mode*. In *Command Mode* whatever you type is interpreted as editor commands. You can then simply enter one or more lines of macro language commands and VEDIT immediately executes the commands.

Besides this interactive Command Mode, VEDIT can execute VEDIT macro language commands in two ways:

- Short sequences of macro language commands can be executed as *keystroke macros*. A sequence of up to 1024 characters can be executed; typical sequences are much shorter.
- Longer sequences of macro language commands called *Command Macros* are typically stored as disk files. These Command Macros can be loaded and executed with {**MISC, Load/Execute Macro**}.

Several Keystroke Macros in the “normal” layout contain macro language commands; one example is <Alt>-<T> (Toggle file). Many keys in the layouts to emulate other editors are implemented as such Keystroke Macros.

The library file **keymac.lib** lists useful Keystroke Macros consisting of macro language commands.

The library file **macros.lib** contains Command Macros in addition to those listed in this chapter.

Once a Command Macro is loaded into VEDIT, it can be executed with {**MISC, Execute Macro**} (<Shift>-<F8>).

Command Macros

Command Macros are sequences of macro language commands stored in the Text Registers. They are typically loaded into the Text Registers from disk files.

There is nothing special about the way Command Macros are stored in Text Registers—there is no difference between Text Registers or even edit buffers that contain text and those that contain Command Macros. It is up to you to keep track of which registers contain text and which contain commands. (It is possible to accidentally execute normal text as a Command Macro, with unpredictable results.)

Command Macros are usually stored in files with a **.vdm** filename extension. There is nothing special about these files—they are normal text files containing the macro commands and (hopefully) descriptive comments.

The VEDIT macro language has a free-form format—commands can occur one per line or many per line.

Loading and Executing Command Macros

To load and execute a Command Macro:

1. Select **{MISC, Load/Execute Macro}**.
2. VEDIT will prompt you for the the name of the Text Register in which to load the macro. Valid register names are **0** through **9** and **A** through **Z**. Type the appropriate letter/digit or simply press <Enter> to use the default register **Z**. Note that if you are running an editor emulation, register **Z** may be in use.
3. VEDIT will prompt for the filename using its standard filename dialog box. Either enter the name of the macro or immediately press <Enter> for “Point & Shoot” file selection. The Point and Shoot “filter” defaults to the **.vdm** files in the “VEDIT Home Directory”, typically **/usr/lib/vedit/*.vdm**, for quick macro selection.

NOTES:



Some Command Macros, such as **print.vdm**, can be loaded and executed from any Text Register; others will only work properly from the default **z** register.



Although our supplied macros usually preserve your edit changes, the fate of the files you are editing depends upon the macro. Some macros will return you to your editing while others have a main menu from which you must exit back to QNX in order to terminate the macro. Therefore, be sure to save your files before running unfamiliar macros.

Once a Command Macro begins running, it often uses additional Text Registers for its own use. Some may be used as “subroutine” macros; others as “string variables”. A Command Macro can write-protect its registers to prevent you from accidentally altering them during block operations.

Properly written Command Macros use at most Text Registers **G** through **Z**, leaving **0** through **9** for your block operations and **A** through **F** as edit buffers for editing additional files. Many macros document their Text Register usage at the beginning.

{HELP, Text Registers} lists which register are currently in use.

Although Command Macros are usually loaded and executed in one operation, as above, it is possible to just load a Command Macro.

To only load a Command Macro:

1. Select **{FILE, Buffer Switch}** (<F4>). At the prompt type the name of the Text Register into which to load the Command Macro.

Note that this also converts the Text Register into an Edit Buffer.

2. Select **{FILE, Insert File}**.

At the prompt enter the full path and filename for the Command Macro. Point and Shoot file selection can also be used.

Some users may prefer to load a Command Macro by entering “Command Mode” and issuing the “+RL” command followed by the name of the macro.

Macro Language in Keystroke Macros

Sequences of macro language commands can be executed as part of a Keystroke Macro. Our default keyboard layout contains several such Keystroke Macros. Emulation layouts contain many Keystroke Macros.

When adding a Keystroke Macro containing macro language commands, follow these guidelines:

- Use **{MISC, Keyboard Layout, Add Keystroke Macro}** to define the macro, you can't use the "Record Macro" function.
- The keystroke macro must begin with **VISUAL EXIT** (<Ctrl>-<E>) to enter Command Mode. When done, the Keystroke Macro will return automatically to normal editing. A final "V" command in the macro language is not needed and should not be used.
- The upper limit for the length of a single Keystroke Macro is 1024 characters. However, complex macros that display menus, etc., should be loaded into Text Registers.
- Macro commands can also be mixed with normal Edit Functions and Menu selections. Use **VISUAL EXIT** to switch to Command Mode. It automatically switches back (to "Visual Mode") when any Edit Function, e.g. **MENU** or **CURSOR UP**, is encountered. When necessary, **ESCAPE** can be used to force the switch back.
- As usual, **{CONFIG, Save to disk}** must be used to make any new Keystroke macros permanent.

Macro Command Examples

The supplied file **keymac.lib** is a library of useful Keystroke Macros, some of which you may want to build into your VEDIT.

The following examples show how simple some useful macro commands can be.

The following Keystroke Macro example duplicates the current line of text on the next line. It also moves the cursor to the beginning of the new line. This saves typing a line of text which is identical or nearly identical to the previous line.

```
[VISUAL EXIT] OL G
```

This Keystroke Macro could also be implemented using **{BLOCK}** operations, but use of the macro language is simpler.

The next example is a Keystroke Macro which moves the cursor to the beginning of the next sentence.

```
[VISUAL EXIT] N/. |S/ F/|F/ -C
```

Auto-Startup & Auto-Execution

Automatic Startup

Upon startup, VEDIT normally searches for the file **veditrc** in the VEDIT Home Directory.

By default, the VEDIT Home Directory is **/usr/lib/vedit**. This may be overridden by setting the environment variable **VEDIT_PATH** to point to another directory.

When emulating another editor, VEDIT searches the same directory for a different file—for example **vi**rc to emulate **vi** or **qedrc** for the **qed** emulation of QNX 2.15 **ed**.

If it finds the file, it loads it into Text Register “Z” and executes it as a Command Macro. The **veditrc** file is optional and no error is given if it is not found.

The **veditrc** file can contain **EO**, **EP**, **ES** and **ET** commands to override the built-in configuration settings.

The command “**YKL filename**” can be used to load a keyboard layout file. This overrides the built-in keyboard layout and Keystroke Macros. For example, to load the keyboard layout file “**BOBS.KEY**” at startup, you would create a **veditrc** file containing the line:

```
YKL BOBS.KEY
```

The system administrator can edit these VEDIT initialization files as required to customize VEDIT for your environment. Any changes made here will affect all users.



/usr/lib/vedit and all files in it should be owned by an administrative account (often, but not necessarily, the overall system administrator) and protected against writing by other users. They are set up this way at installation. Allowing the general public to write to these files allows anyone to make arbitrary system-wide changes in VEDIT behavior. Some possible changes could create a serious security risk.

Sequence of Files

VEDIT looks for its first initialization file—**veditrc**, **vi**rc or **qedrc**—in the “VEDIT Home Directory”, typically **/usr/lib/vedit**.

With the default setup, VEDIT next searches for a user-specific initialization file in the user's home directory and executes that file as a macro if found. The filename searched for is **.veditrc** for "straight" VEDIT or **.vedit.virc** or **.vedit.qedrc** for the emulations.

The search for these user-specific files could be disabled or modified by your system administrator making changes to the startup VEDIT initialization file.

Filenames beginning with a period are not normally shown by the **ls** command or matched by shell pattern-matching expressions such as *****. They are shown by **ls -a** and matched by expressions with an explicit period such as **.***. They can still be edited, for example with **vedit .veditrc**, and manipulated with other QNX commands. Consult your *Utilities Guide* sections on **ls** and **sh** for more detail on this.

The user's initialization file can optionally include the lines:

```
:YKL qed.key
:YJL qed.cfg
:ED qed.ini
(.rv=1)[RN(z) qed.ini]
[RN(z) /usr/lib/vedit/qed.vdm]
```

to cause VEDIT to search for **vedit.ini** in the directory from which it is invoked and execute that file as a macro if found.

The first two lines above just read files. The **:ED** command tests for the existence of the file **qed.ini**. The **(.rv=1)** tests the return value from the **:ED** and executes one of the two **RN(z)** commands. Those commands *run* a file as a macro using text register **z**.

If **vedit.ini** in the current directory is used, then you can have different **vedit.ini** files in different directories to make VEDIT behave differently for different applications. For example, in a directory used for word processing, **vedit.ini** could set the right margin for word wrap while in a directory used for C programming, **vedit.ini** could enable Auto-Indent mode and disable word wrap.



System administrators and other holders of privileged accounts should not use this technique. If a privileged user starts VEDIT and executes **vedit.ini** in the current directory, then *whatever* commands are in **vedit.ini** are executed *with all that user's privileges*. There is a risk of a user creating a “Trojan horse” **vedit.ini** which abuses these privileges to violate system security.

Auto-Execution

VEDIT's “*auto-execution*” lets you specify a different Command Macro to execute in place of, or in addition to, the **veditrc** file. Auto-execution is controlled with two startup switches.

- i** *execfile* Execute **execfile** on startup *instead of* the normal **veditrc** file.
- x** *execfile* Execute **execfile** on startup *after* the normal **veditrc** file.

In either case, **execfile** is searched for in the VEDIT Home Directory, is loaded into Text Register **z** and is executed as a Command Macro. If no filename extension is given **.vdm** is assumed.

For example, the command to print the file **chapter2** using the supplied **print.vdm** macro is:

```
vedit -x print.vdm chapter2
```

In this case, VEDIT will first search for **/usr/lib/veditrc** and execute it if found. **print.vdm** will then be loaded and executed.

You can also specify macro commands to be executed on the command line with the **-c** startup option.

- c** *command* The macro language commands *command* are executed upon startup.



A *command* containing spaces, or other characters which the shell will interpret, must be quoted. Usually the simplest way to do this is to enclose the command in single quote (') characters. See the *Utility Guide* shell section for alternatives and complications.

For example, the command to start up the editor on line 451 of the file **datafile** is:

```
vedit -c 450L datafile
```

When both the **-c** and **-x** options are used, the leftmost one will execute first; however, both execute *after* any **veditrc** file.



You can disable the **veditrc** auto-startup feature on an occasional basis by using **-i** followed by a non-existent file.

Auto-execution via
Shellscripts

You can use QNX “shellscripts” (or shell aliases if your shell supports this feature) to quickly invoke commonly used VEDIT macros.

For example, the shellscript:

```
for i
do
    vedit -x print.vdm $i
done
```

will print each of its argument files in turn using VEDIT and the **print.vdm** macro.

Another example is the one-liner:

```
vedit -x wildfile.vdm
```

which could be either a shellscript or an alias. It invokes VEDIT and runs the **wildfile.vdm** macro.

Print - Print Macro

The macro **print.vdm** can be used as an alternative to the basic print commands since it also prints the filename, date and page number at the top of each page. It also skips page perforations and indents the text from the left paper edge. This makes it ideal for printing source code modules and other text files.

To print a file using **print.vdm** from the QNX prompt:

1. Give the command:

```
vedit -x print.vdm filename
```

To run **print.vdm** from within VEDIT:

1. Select **{MISC, Load/Execute Macro}**.
2. At the “Register Name” prompt type the name of any desired Text Register. Many users choose register **P**.
3. At the “Enter Filename” prompt either enter “**print**” or press <Enter> for “point and shoot” file selection.

The entire file should begin printing. To stop the printing before it is done press **CANCEL** (<Ctrl>-<C>).

When done, **print** gives you the choice of printing another file, returning to your editing or leaving VEDIT.

If you remain in VEDIT, you can start up the print macro again (assuming it is still loaded) with **{MISC, Execute Macro}** (<Shift>-<F8>). At the “Register Name” prompt type the same register name as before, typically “P”.

If you frequently use **print.vdm** from within VEDIT, you can automatically load into Text Register **P** by placing the command **+RL(P) print.vdm** into the **veditrc** auto-startup file.

Much more sophisticated formatters can be written with VEDIT macros. **print.vdm** is intended as a macro example which is relatively easy to understand and expand. Possible enhancements include line numbering, header and footer messages and more.

Wildfile - Multi-file Processing

With **wildfile.vdm** it is easy to perform a search, search and replace or even run another macro on an entire group of files. The group of files may be specified using the wildcards **?** and *****.

The primary use of **wildfile** is to search for all occurrences of a word (variable name, etc.) in a group of files. For example, you might want to view all occurrences of the word **printf** in all the **.c** files on disk.

Example - To view all occurrences of **printf** in all **.c** files:

1. Invoke **wildfile** with the command:

```
vedit -x wildfile.vdm
```

2. At the filename prompt enter ***.c** and press <Enter> twice.
3. At the next prompt type **S** to specify that you want perform a search. At the prompt for the search string enter **printf**.
4. At the next prompt type **V** to view each occurrence in Visual Mode. At this point the screen will look as follows:

```

Command Mode      "H"=Help
*****
*                  *
*          UEDIT PLUS          *
*      Multi-file Macro Driver      *
*                  *
*****

Enter the names of the files you want to operate on and <Enter>.
(The wildcards '?' and '*' may be used).

Press <Enter> twice when done.

Enter Filename: *.c
Enter Filename:

Execute a [C]ommand, [M]acro or [S]earch? (C/M/S):

Enter search string (and Enter): printf

[D]isplay line or enter [V]isual Mode? (D/V):
Press any key to begin operation..._

```

wildfile can also perform global replacements on many files. For example, you might have misspelled “parallel” as “parallel” in all the **.txt** files on your disk.

To correct a spelling error in all ***.txt** files:

1. Invoke the **wildfile** with the command:

```
vedit -x wildfile.vdm
```

2. At the filename prompt enter ***.txt** and press <Enter> twice.
3. At the next prompt type **R** to specify that you want perform a (search and) replace. Then enter the desired search and replacement strings.
4. At the “[D]isplay line or enter [V]isual Mode?” prompt type **D** if you only want to have each altered line displayed on the screen, or type **V** if you want to enter the normal “Visual” editing mode after each replacement is made.

The **wildfile.vdm** macro can also be used to run a second macro on a group of files. For example, the **print.vdm** macro could be run on all of your **.txt** files.

You may find yourself using **wildfile.vdm** often!

Compare - File Comparison macro

compare.vdm can compare two text files of arbitrary size. One file, called the “*active file*” can also be edited while it is being compared to the “*template file*”. The files can be merged by copying blocks of text from the “template” file to the “active” file. You will find many uses for this file comparison including merging the work done by several people, or simply determining the differences between two versions of a file.

compare uses windows to give you a split-screen comparison of the two files. Remember that the macro can only save changes you make to the “active file”, not to the “template file”.

To perform the file comparison give the QNX command:

```
vedit -x compare.vdm
```

compare then displays the active file with the cursor at the first position of the active file that differs from the template file. Note that **compare** handles the case where entire lines have been added or deleted. You can edit the active file in any way you wish. If you selected “windows”, the template file will also be displayed with its own cursor.

[illegible]

Although **compare** does not save changes to the template file, *you can copy text from the template file to the active file* using Text Registers “0” through “9”. For this you must switch back and forth between the files using one of these methods:

- With the default keyboard layout, press <Alt>-<T> to toggle between the two files.
- Select {**FILE, Buffer Switch**} (<F4>). Type **P** to switch to the template file; press <Enter> to switch back to the active file.
- If both files are displayed in windows, you can click your mouse in the desired window.
- Press **ESCAPE**.

The following menu will appear:

```
BLOCKS.TXT                                Command Mode                                "H"=Help
File comparison interrupted. Please select one of the following options:
1) Examine template file                    3) Resume, no realignment
2) Examine active file                      4) Realign template & resume
5) Stop. Get termination options menu

Enter Option:
```

Options 1 and 2 let you switch to the desired file.

When you press **VISUAL EXIT** while examining the active file, **compare** examines the 24 characters following the cursor and attempts to find them in the template file; if this alignment is successful, it resumes the file comparison. This is the same as selecting option 4 from the menu.

On the other hand, when you press **VISUAL EXIT** while examining the template file, **compare** immediately resumes the file comparison from the current cursor positions. This is the same as selecting option 3 from the menu.

compare attempts to align the template file with the active file by looking within the template file for the 24-character string following the active-file cursor. If it cannot find a match within the twenty lines preceding the template cursor nor within the hundred lines following, it issues the error: "Unable to realign template file".

Realignment failure is most likely due to the active file cursor being positioned where the files are not identical for the next 24 characters. It could also be due to the cursor having been moved too far forward or backward. To continue the comparison, reposition the active file cursor and press **VISUAL EXIT**.

Occasionally, **compare** will be unable to realign the files. To continue the file comparison you must then manually realign the cursor in both files and select option 3 from the above menu.



The **compare** macro does not automatically save any changes to the “template” file. However, you can manually save changes by switching to the template file and selecting **{FILE, Save and Continue}**.

Sort - Mailing List Sort

The **sort** macro alphabetically sorts a simple mailing list consisting of address lines separated by one or more blank lines. The sort is based on the first address line, assumed to be a name. For example, the following list could be sorted:

```
Scott, Charles
3219 Space Ct.
Albany, NY 14311
-(305) 321-7654
-Broadcast Producer
```

```
Burnett, Tammie
642 Sunset Blvd.
Miami, FL 32103
-Travel Consultant
```

```
Mathews, Lee
236 Bluelake Dr.
Marquette, MI 48123
-(313) 123-4567
-Basketball Player
```

To run the **sort** macro issue the command:

```
vedit -x sort.vdm
```

sort prompts for the filename of the file to be sorted and for the name of the file to contain the sorted output. Just press <Enter> if the latter is the same as the former. **sort** will then sort the list, save the new list on disk and return to QNX.

It takes a few seconds per address to sort a medium sized file.

Chapter 5

Menu Reference

This chapter provides a reference for specific menu items and operations. An overall description of how to use the VEDIT menu system can be found in Chapter 3 (Editing Guide).

This chapter lists the menus in the order in which they appear on the top menu line. Introductory information pertaining to each menu is given. Also included are the “Hot-Key” assignments in the default keyboard layout which may be used to directly access each menu and many menu items.

A description of each submenu item is divided into the following sections:

Menu Item	The name of the menu item as it appears on the screen.
Keystroke Equivalent:	If the default keyboard layout has a “hot key” or other keystroke equivalent to selecting this item, it is listed here. Use {MISC, Keyboard Layout, Display Layout} to list all available keystroke operations for your keyboard layout.
Description:	A description of the menu item and how to use it. This often includes step-by-step instructions for performing common operations.

Notes: Related comments, suggestions and warnings.

See Also: List of related menu items, edit operations and other supporting references.

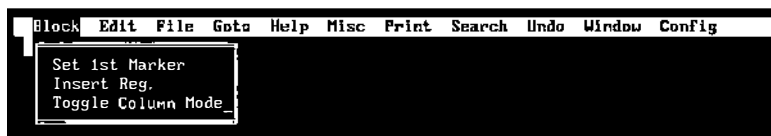
Block Menu

Keystroke Equivalent: <Alt>-, This is a Keystroke Macro.

Introduction: The Block Menu contains items for manipulating blocks of text. These include marking a block (highlighting the desired text), copying or moving a block, saving or inserting a block and other related items.

Many of the Block Menu items are not usable until one or both block markers are set. Therefore the Block Menu only displays those items which are currently usable. The Block Menu may appear in one of three forms.

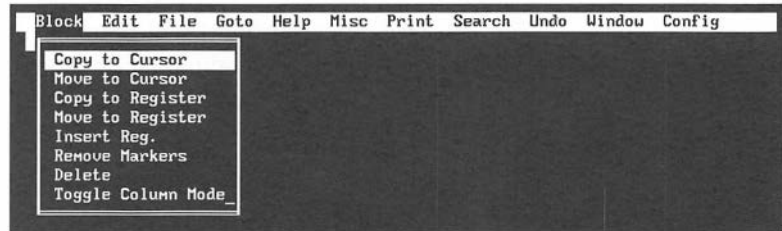
Before you set either block marker:



After you set the first block marker:



When both block markers are set:



Set Marker

Keystroke Equivalents:

Marks the beginning and end of a “character” block of text.

<F9>, this is a Keystroke Macro. Alternatively, **BLOCK COPY** can be used to set block markers.

Description:

Before you can copy, move, delete or otherwise manipulate a block of text, you need to set “*block markers*” to select the area of text.

To mark a block of text (assuming no block markers are set):

1. Position the cursor on the first character to be included in the block. (If you prefer, you can mark the end of the block first.)
2. Set the first block marker with **{BLOCK, Set Marker}** (<F9>). The message “I-END” will be displayed on the status line.
3. As you now move the cursor to the end of the block, intervening text will be highlighted to define the block. (The cursor can be moved in either direction, i.e. you can mark the end of the block first and then mark the beginning of the block.)
4. Set the second marker by again selecting **{BLOCK, Set Marker}** (<F9>). The message “BLOCK” will be displayed on the status line.

After setting the second marker, you can freely move the cursor anywhere in the file without affecting the highlighted block, even if doing so moves the highlighted text off the screen.

You can change the size of a highlighted block after setting the marker(s). If only the first marker is set, use **{BLOCK, Swap Ends}** to re-adjust the first marker. If both markers are set, **{BLOCK, Set Marker}** will re-adjust the nearest marker to expand or contract the highlighted block of text.

To remove (clear) the block markers at any time:

Press **CANCEL** (<Ctrl>-<C>) or select **{BLOCK, Remove Markers}**.

Pressing **CANCEL** is usually simpler.

Notes: Many block operations can be performed after setting only the first marker; the cursor position acts as the second marker.

You must mark the end of the block with the cursor one character PAST the last character to be included in the block. This may not make sense at first, but once you are used to it, it simplifies block operations. See “Block Operations” in Chapter 3 (Editing Guide) for a more complete explanation.

Having a block highlighted with both markers affects search operations and lets you restrict the search to the highlighted text. See “Searching within a Block” described under **{SEARCH, Search}** in this chapter.

The block markers may also be set using the mouse.

See Also: “Block Operations” in Chapter 3 (Editing Guide).
{BLOCK, Copy to Cursor}, **{BLOCK, Set Column Marker}**
T-REG COPY, T-REG MOVE
BLOCK COPY, BLOCK MOVE

**Set Column
Marker**

Keystroke, Edit Function
Equivalent:

Marks the beginning and end of a “columnar” block of text.

<Ctrl>-<N>

Description: “Set Column Marker” is identical to “Set Marker” except that it selects (and highlights) “columnar blocks”. In a columnar block, only text which is located within a rectangle whose opposite corners are defined by the first block marker and the cursor (or second marker) is selected.

Notes: Selecting **{BLOCK, Set Column Marker}** after **{BLOCK, Set Marker}** will change the block from a “character” to a “columnar” one. The opposite is also true.

By highlighting a columnar block with both block markers, you can restrict a search or search & replace operation to the selected columns.

When highlighting a columnar block, the cursor can be moved past the end of a line. This action is similar to cursor mode 4 and makes it easier to select a column of text.

After copying a columnar block to a Text Register, VEDIT remembers that the register contains a column block and will, in interactive use, insert it differently than a character block. Note that this does not hold for the macro language.

The heading “Block Operations - Columnar Blocks” in Chapter 3 (Editing Guide) describes columnar blocks in detail. This includes how VEDIT pads and trims spaces at the ends of lines and deals with Tab characters.

See Also: “Block Operations” in Chapter 3 (Editing Guide).
{BLOCK, Set Marker}

Copy to Cursor

Copy the highlighted block of text to the current cursor position.

Keystroke Equivalent: <Ctrl>-<F9>, identical to **[BLOCK COPY]**.

Description: “Copy to Cursor” copies block of text from one place in your file to another. It does not delete or otherwise affect the original block.

To copy a block of text:

1. Highlight the desired text using both block markers.
2. Move the cursor to the location in the file where you would like the text copied. The text will be copied in front of the cursor.
3. Select **{BLOCK, Copy to Cursor}** or press **BLOCK COPY** (<F9>).

Notes: This function normally clears the block markers. However, by setting the configuration parameter **{CONFIG, Emulation, Auto-Cancel Block Markers}** to “No”, the block will be highlighted in its new location.

If you need to repeatedly copy the same block to multiple locations in your file or files, it is easier to copy the block to a Text Register and then insert the register where needed.

You can also copy blocks using the mouse.

See Also: “Block Operations - [BLOCK COPY] and [BLOCK MOVE]” in Chapter 3 (Editing Guide).
{BLOCK, Set Marker}, **{BLOCK, Move to Cursor}**
{BLOCK, Copy to Register}
BLOCK COPY, **BLOCK MOVE**

Move to Cursor

Move the highlighted block of text to the cursor position.

Keystroke Equivalent: <Alt>-<F9>, identical to **[BLOCK MOVE]**.

Description: This item is similar to **{BLOCK, Copy to Cursor}** except that the original highlighted text is deleted after it is copied to the new location. In more common terms, it’s as if you “cut” the text out of the original location and “pasted” it into the current cursor location.

To move a block of text:

1. Highlight the desired text using both block markers.
2. Move the cursor to the new location for this text. The text will be moved to just in front of the cursor.
3. Select **{BLOCK, Move to Cursor}** or press **BLOCK MOVE** (<Alt>-<F9>).

Notes: The notes for {**BLOCK, Copy to Cursor**} also apply here.

You can also move blocks using the mouse.

See Also: “Block Operations - **BLOCK COPY** and **BLOCK MOVE**” in Chapter 3 (Editing Guide).

{**BLOCK, Set Marker**}, {**BLOCK, Copy to Cursor**}
BLOCK COPY, BLOCK MOVE

Copy to Register

Save a copy of the highlighted block of text in a “Text Register”.

Keystroke Equivalent: <Ctrl>-<F10>, identical to **T-REG COPY**.

Description: A “*Text Register*” is a temporary holding area for text you would like to use later. Similar to a memory on a calculator, once text is saved in a Text Register, it can be used again and again. Text you save stays in the register until you either save a different block of text or exit VEDIT.

To save text into a Text Register:

1. Highlight the desired text. You can either set both markers or use the cursor as the second marker.
2. Select {**BLOCK, Copy to Register**} or press [T-REG COPY] (<F10>). A dialog box will prompt you for the name of the desired Text Register; valid register names are “0” through “9” and “A” through “Z”.
3. Press the appropriate letter/digit to select a Text Register. The status line message “TEXT” now indicates that something is stored in at least one Text Register.

You can optionally append the highlighted block to any existing Text Register contents by preceding the register name with “+”; or insert the highlighted block in front of any existing Text Register contents by preceding the register name with “-”.

HINT:



When prompted for a Text Register name, you can press any function key or <Enter> to select the default register “0”. For example, you can simply press the key for **T-REG COPY** twice to copy the highlighted block to register “0”.

Notes:

The word “TEXT” is normally displayed on the status line whenever text is stored in a Text Register. However, “1-END” will replace “TEXT” when the first block marker is set and “BLOCK” will be displayed when both are set.

This function always removes the block markers.

Text Register contents are lost when you exit VEDIT.

There is a limit to the amount of text which can be saved in the Text Registers. This is usually between 40,000 and 50,000 characters, but can be much less if VEDIT has limited memory available to it.

If memory is limited or the size of the block is very large, you may see one of two error messages:

NOT ENOUGH MEMORY FOR OPERATION

FILE POSITION NOT IN MEMORY

In either case, there is not enough memory to save all of the highlighted text in a Text Register. Try working with several smaller pieces of text.

See Also:

“Block Operations - Text Registers” in Chapter 3 (Editing Guide).
{BLOCK, Set Marker}, **{BLOCK, Copy to Cursor}**
T-REG COPY, **T-REG MOVE**

Move to Register

Move the highlighted block of text into a “Text Register”.

Keystroke Equivalent:

<Alt>-<F10>, identical to **T-REG MOVE**.

Description:

This item is similar to **{BLOCK, Copy to Register}** except that the highlighted text is deleted from your file as it is placed in the Text Register.

To move text into a Text Register:

1. Highlight the desired text. You can either set both block markers or use the cursor position as the second marker.
2. Select **{BLOCK, Move to Register}** or press **T-REG MOVE** (<Alt>-<F10>).
3. Select the desired Text Register in the same way as for **{BLOCK, Copy to Register}**.

See Also: Refer to the description of **{BLOCK, Copy to Register}** for more information.

“Block Operations - Text Registers” in Chapter 3 (Editing Guide).
{BLOCK, Set Marker}, **{BLOCK, Move to Cursor}**
T-REG COPY, **T-REG MOVE**

Insert Register

Insert the contents of a Text Register at the current cursor position.

Keystroke Equivalent: <F10>, identical to **T-REG INSERT**.

Description: “Insert Register” inserts the contents of a Text Register at the current cursor location. If there is no text in the Text Register, selecting this item has no effect.

To insert the contents of a Text Register:

1. Position the cursor at the desired location in your file; the register contents will be inserted just before the cursor.
2. Select **{BLOCK, Insert Register}** or press **T-REG COPY** (<F10>). A dialog box will prompt you for the name of the desired Text Register; valid register names are “0” through “9” and “A” through “Z”.
3. Press the appropriate letter/digit to select a Text Register. The inserted contents of the Text Register should now appear.

HINT:



When prompted for a Text Register name, you can press any function key or <Enter> to select the default register 0. For example, you could press the <F10> key twice to insert register 0.

Notes: This item always inserts text regardless of the “Insert Mode” status for normal editing.

You can insert a Text Register even when a block is currently highlighted.

If desired, set the configuration parameter **{CONFIG, Emulation, Advance cursor past block insert}** to “No” to have the cursor remain at the beginning of the inserted text.

See Also: **{BLOCK, Copy to Register}, {BLOCK, Move to Register}**
T-REG COPY, T-REG MOVE

Delete Delete the highlighted block of text.

Keystroke Equivalent: (none)

Description: Any size block of text may be deleted; however, “Undo” may not be able to restore very large text deletions.

To delete a block of text:

1. Highlight the desired text. You can either set both markers or use the cursor as the second marker. You can highlight either a “character” or a “columnar” block.
2. Select **{BLOCK, Delete}**. The block will be deleted.

To insert the just deleted block of text somewhere else:

1. Position the cursor where you want the block re-inserted.
2. Select **{UNDO, Deletion}**.

Notes: If the block being deleted is too large for “Undo” to restore it, you will be prompted for verification whether to still want to delete the block.

You can effectively also “delete” a block of text by moving it into a Text Register.

See Also: **{BLOCK, Set Marker}, {BLOCK, Move to Register}**
{UNDO, Deletion}

Write to Disk

Write the highlighted block of text to a disk file.

Keystroke Equivalent:

(none)

Description:

A block of any size may be saved to disk. You can use this feature for dividing a file into several pieces or saving parts of your file for use in other files.

Unlike saving text in a Text Register, text written to disk will be available after you leave VEDIT.

To save a block of text to disk:

1. Highlight the desired text. You can either set both markers or use the cursor as the second marker. You can highlight either a “character” or a “columnar” block.
2. Select **{BLOCK, Write to Disk}**. VEDIT will prompt for the desired filename using its standard filename dialog box.
3. Either enter the filename or immediately press <Enter> for “Point & Shoot” file selection.

Remove Markers

Remove any existing block markers.

Keystroke Equivalent:

(None), however, CANCEL (<Ctrl>-<C>) is usually used to remove markers.

Description:

You can remove (clear) the block markers at any time; this also removes the highlighting. Normally, most block operations clear the block markers when completed. However, if you have configured **{CONFIG, Emulation, Auto-Cancel Block Markers}** to “No”, you will have to explicitly remove the markers after many block operations.

To remove (clear) the block markers at any time:

1. Press CANCEL (<Ctrl>-<C>) or select **{BLOCK, Remove Markers}**.

Notes:

This menu item is primarily only selected with a mouse, because it is usually more convenient to simply press CANCEL (<Ctrl>-<C>).

This item is also available in the Escape Menu.

To remove the block markers with a mouse, press both mouse buttons—this performs the **CANCEL** function.

See Also: **{BLOCK, Set Marker}**
CANCEL

Swap Ends

Swap the position of the cursor with the position of the first block marker.

Keystroke Equivalent: (none)

Description: “Swap Ends” exchanges the positions of the first block marker with the current cursor position. This lets you re-adjust either end of the highlighted text. It is also useful when you forget precisely where the beginning of a large block is after it has scrolled off the screen.

To check and/or adjust the beginning of a highlighted block of text:

1. Select **{BLOCK, Swap Ends}**. The cursor and first marker positions will be exchanged.
2. If necessary, move the cursor to adjust the block.
3. Select **{BLOCK, Swap Ends}** again. You can now adjust this end of the block.

Notes: This function is only available when the first and not the second block marker have been set. If both markers are set, **{BLOCK, Set Marker}** will re-adjust the nearest marker.

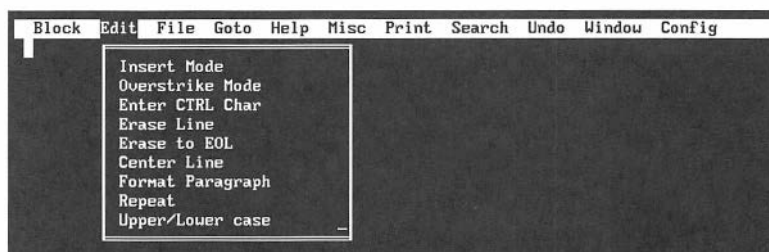
If a block is extremely large, **{BLOCK, Swap Ends}** can result in a significant amount of disk activity. This is perfectly normal.

See Also: **{BLOCK, Set Marker}**

Edit Menu

Keystroke Equivalent: <Alt>-<E>, This is a Keystroke Macro.

Introduction: The Edit Menu provides an assortment of items related to typing and making quick changes to your file.



Insert Mode

Explicitly selects “Insert” mode for entering new text.

Keystroke Equivalent: (None), however, **INSERT TOGGLE** (<Ins>) is normally used to toggle between “Insert” and “Overstrike” mode.

Description: In Insert Mode, characters you type are inserted in front of the character at the cursor. The message “INS” is displayed on the status line.

{EDIT, Insert Mode} explicitly puts VEDIT into Insert Mode. Unlike **INSERT TOGGLE** (<Ins>), which toggles between Insert and Overstrike Modes, this item can be used in a Keystroke Macro to ensure that Insert Mode is on.

Notes: Insert/overstrike modes only affect the typing of new text. Other types of insertions, such as **{BLOCK, Insert Register}**, are always insertions and never overlay the existing text.

See Also: **{EDIT, Overstrike Mode}**
INSERT TOGGLE

Overstrike Mode

Explicitly selects “Overstrike” mode for entering new text.

Keystroke Equivalent: (None), however, **INSERT TOGGLE** (<Ins>) is normally used to toggle between “Insert” and “Overstrike” mode.

Description: In Overstrike Mode, characters you type replace any existing text at the cursor position. The message “INS” is not displayed on the status line.

{EDIT, Overstrike Mode} explicitly puts VEDIT into Overstrike Mode. Unlike **INSERT TOGGLE** (<Ins>), which toggles between Insert and Overstrike Modes, this item can be used in a Keystroke Macro to ensure that Insert Mode is off.

See Also: **{EDIT, Insert Mode}**
INSERT TOGGLE

Enter CTRL Char

Enters the next key press literally into the text.

Keystroke Equivalent: <Ctrl>-<Q>, This is identical to **ENTER CTRL**.

Description: Each time you press a key, VEDIT determines if that key represents an editing command or is simply a character to be used in the text. At times you may want to enter “control characters” into the text which VEDIT normally interprets as an editing command. This function forces VEDIT to interpret the next key press as a character to be added to your text.

Certain control characters can be useful in your text. Most notably, “Bell” (<Ctrl>-<G>), “Line-Feed” (<Ctrl>-<J>), “Form-Feed” (<Ctrl>-<L>) and “Escape” (<Esc>) are frequently used. To enter the control characters, you must precede them with **ENTER CTRL**.

For example, printing a file containing Form-Feed characters will cause the printer to start a new page when each Form-Feed character is encountered.

To insert a control character into your file:

1. Make sure you are in Insert Mode—**INS** should be displayed on the status line. Otherwise the control character will overstrike any existing character.
2. Position the cursor where the control character is to be inserted.
3. Select **{EDIT, Enter CTRL}** or press **ENTER CTRL** (<Ctrl>-<Q>). VEDIT will prompt you for the control character.

4. Type the desired control character. For example, to insert a `<Ctrl>-<C>`, hold down the `<Ctrl>` key while you type `C`.

OR

4. To enter a control or graphics character by its ASCII value, press and release the `<Alt>` key and then type the desired value as two hexadecimal digits.

Notes: This item is rarely used, because it is usually more convenient to simply press **ENTER CTRL** (`<Ctrl>-<Q>`). However, this is a logical item to have in the **{EDIT}** menu for new users.

Characters may also be entered by ASCII value using **{MISC, ASCII Table}**.

This is the only way to enter a value of “00” (i.e `<Ctrl>-<@>`) into the text, because the IBM PC keyboard does not generate a value of “00”.

See Also: “Keyboard Characters and Screen Display” in Chapter 3 (Editing Guide).
{MISC, ASCII Table}, **{CONFIG, Misc, Screen Output processing}**
ENTER CTRL

Erase Line

Deletes (erases) the entire line.

Keystroke Equivalent: `<Ctrl>-<X>`, identical to **ERASE LINE**

Description: “Erase Line” deletes the entire line of text on which the cursor is located. It also deletes the **newline** character at the end of the line and closes up the line on the screen.

To erase an entire line of text:

1. Place the cursor anywhere on the line to be erased.
2. Select **{EDIT, Erase Line}** or press **ERASE LINE** (`<Ctrl>-<X>`). After the line is erased, the cursor will be located on the same column of the following line (if any).

Notes: This menu item is primarily only selected with a mouse, because it is usually more convenient to simply press **ERASE LINE** (`<Ctrl>-<X>`).

You can use {**UNDO, Deletion**} to re-insert deleted lines in reverse order or in a different location. Up to six lines can be re-inserted in this way.

See Also: **ERASE LINE**

Erase EOL

Deletes (erases) text from cursor position to end of line.

Keystroke Equivalent: <Ctrl>-<Z>, identical to **ERASE EOL**

Description: “Erase EOL” deletes all characters from the current cursor position to the end of the line. It does not delete the **newline** character(s) at the end of the line.

To erase from the cursor position to the end of the line:

1. Place the cursor on the first character to be erased.
2. Select {**EDIT, Erase EOL**} or press [**ERASE EOL**] (<Ctrl>-<Z>). The cursor will not move.

Notes: This menu item is primarily for mouse users. On the keyboard, just press **ERASE EOL** (<Ctrl>-<Z>).

See Also: **ERASE EOL**

Center Line

Center the current line.

Keystroke Equivalent: (none)

Description: “Center Line” adds just enough space to the beginning of a line to center the line between column 1 and the right margin.

To center a line of text:

1. Make sure that the right margin is set correctly. When the right margin is set, the status line displays “RM:” and its value. If necessary, set the right margin with {**CONFIG, Word Processing, Right margin**}.
2. Select {**EDIT, Center Line**}.

This will center the current line and advance the cursor to the next line so that you can easily repeat this process to quickly center several consecutive lines.

Notes: Unlike other functions which indent with the optimal number of tabs and spaces, “Center Line” only adds space characters as needed.

To center a number of lines or an entire paragraph, precede this function with **REPEAT** and the desired count.

See Also: {CONFIG, Word Processing, Right Margin}

Format Paragraph

Formats a paragraph between the left and right margins.

Keystroke Equivalent: <Ctrl>-<F>, This is a Keystroke Macro

Description: VEDIT can “format” a paragraph to fit it between the current left and right margins. To do this, VEDIT may move one or more words from the end of a line to the beginning of the next line or borrow words from the beginning of the following line to fill out a line. In the process, the number of lines in the paragraph may change. VEDIT can optionally add extra spaces between words to “justify” (straighten) the right edge of the paragraph.

To format (or re-format) a paragraph:

1. Make sure that the right margin is set correctly. When the right margin is set, the status line displays “RM:” and its value. If necessary, set the right margin with {CONFIG, Word Processing, Right margin}.
2. Make sure the left margin is set if you want the entire paragraph indented. The left margin is often set with **INDENT** (<F8>) and **UNDENT** (<F7>).
3. Place the cursor anywhere in the paragraph and select {EDIT, Format Paragraph} (<Ctrl>-<F>). After formatting, the cursor is advanced to the beginning of the next paragraph so that you can easily format several consecutive paragraphs.

Notes: Blank lines *must* be used to separate paragraphs from another; just an indented line is not enough. Otherwise, formatting paragraphs will combine several paragraphs into one!

When formatting paragraphs, VEDIT attempts to maintain double spaces between sentences, i.e. after “.”, “!” and “?”.

See Also: “Formatting and Justifying Paragraphs” in Chapter 3 (Editing Guide)
{**CONFIG, Word Processing**} submenu
INDENT, UNIDENT

Repeat

Repeat next edit operation.

Keystroke Equivalent: <Ctrl>-<R>, identical to **REPEAT**

Description: “Repeat” repeats the following edit operation a number of times. Any edit operation, such as typing in a character, pressing an Edit Function, selecting an item from a menu or even executing a Keystroke Macro, can be repeated.

To repeat an edit operation:

1. Select {**EDIT, Repeat**} or press **REPEAT** (<Ctrl>-<R>). You will see the following prompt:

Repeat Count: 4

2. Specify how many times to repeat the upcoming operation. You can do this in several ways. You can accept the default value by doing nothing; enter the desired repeat count; or repeatedly press **REPEAT** (<Ctrl>-<R>), quadrupling the value with each press. **DO NOT PRESS** <Enter>.
3. Now that the desired repeat count is displayed, simply press the Edit Function or displayable character you want repeated. You can also repeatedly execute items from the main menu. One thing you can not do is enter a numeric character.

For example, to create the top of a box, you could press **REPEAT**, “50” and “*”.

Notes: The maximum repeat count is 256.

This menu item is primarily only selected with a mouse, because it is usually more convenient to simply press **REPEAT** (<Ctrl>-<R>).

When selecting menu items to repeat, select the items by typing the appropriate selection letters. Attempting to select items with the cursor keys will confuse the process and may lead to undesired results.

Because of the way VEDIT updates the screen, only the final screen may be shown when using **REPEAT**.

Since some operations may take a bit of time to perform, you may notice some delay when they are repeated many times.

To abort a lengthy “Repeat” press **CANCEL** (<Ctrl>-<C>).

A common mistake is to press <Enter> following the repeat count. Simply select **{UNDO, Edit}** (<Alt>-<F7>) to undo the mistake and try again.

See Also: **REPEAT**

Lower/Upper case (Submenu)

Keystroke Equivalent:

Introduction:

This submenu lets you change the case of an individual character (limited use) or all characters in a highlighted block (very useful).

(none)

Selecting this item displays the following submenu:



See Also: **{CONFIG, Programming, Lower/Upper case key conversion}**

Upper Case

Keystroke Equivalent:

Description:

(none)

“Upper Case” forces a lower case letter to upper case. Other characters are not affected.

If the cursor is in a highlighted block of text (both markers must be set), the entire block will be forced to upper case and the cursor will not move. Otherwise, only the character at the cursor will be forced to upper case and the cursor will advance to the next character.

Lower Case

Keystroke Equivalent:

(none)

Description: “Lower Case” forces an upper case letter to lower case. Other characters are not affected.

If the cursor is in a highlighted block of text (both markers must be set), the entire block will be forced to lower case and the cursor will not move. Otherwise, only the character at the cursor will be forced to lower case and the cursor will advance to the next character.

Switch Case

Keystroke Equivalent: (none)

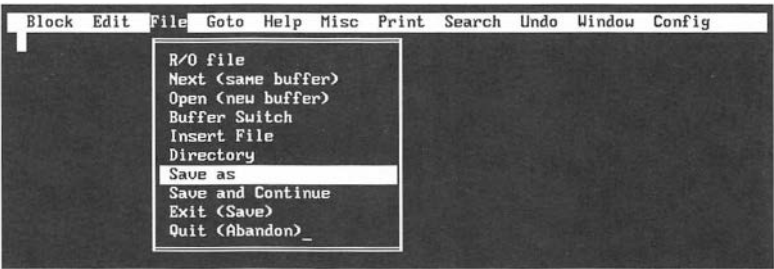
Description: “Switch Case” changes a lower case letter to upper case and an upper case letter to lower case. Other characters are not affected.

If the cursor is in a highlighted block of text (both markers must be set), the entire block will have its case switched and the cursor will not move. Otherwise, only the character at the cursor will have its case switched and the cursor will advance to the next character.

File Menu

Keystroke Equivalent: <Alt>-<F>, This is a Keystroke Macro.

Introduction: The File Menu contains a variety of file related operations. These include selecting a file for editing, inserting text of another file into your current document, saving and exiting, and multi-file editing.



R/O File

Determines whether the current file can be altered or just viewed.

Keystroke Equivalent: (None)

Description: This item normally displays “No” to indicate that the current file can be altered. After selecting this item and toggling it to “Yes”, the file is considered a “Read-only” file which can be viewed but not altered. This is convenient when you want to ensure that a critical file is not accidentally altered.

If you edit a file for which you do not have QNX write permission, or a file locked for exclusive write by another application (for example, another VEDIT session), then this item will be automatically set to “Yes” and you won’t be able to change it to “No”.

After starting up VEDIT in “browse mode” with the **-b** option, all files that you edit are considered “read-only” files which cannot be altered. This item will read “Yes” and you won’t be able to change it to “No”.

Notes: Some versions of VEDIT perform disk buffering differently for “read-only” files than for normal files. Therefore, after you toggle this item to “Yes” and move around in the file, you may not be able to change it back to “No”.

See Also: “Starting VEDIT” in Chapter 3 (Editing Guide).

New (same buffer)

Select a file for editing.

Keystroke Equivalent: <Alt>-<N>, This is a Keystroke Macro.

Description: With VEDIT, you may edit one or more files at a time. Each file is edited in one of the 37 “edit buffers”. When you first enter VEDIT, the main edit buffer named @ is initially active. Use “New File” when you want to edit another file in the same (current) buffer. Use “Open File” when you want to simultaneously edit another file in the next available edit buffer.

To edit another file (in the same buffer):

1. Select **{FILE, New File}** (<Alt>-<N>). If you currently have a file loaded and have made changes to that file, you will be prompted whether the current file should be saved or abandoned before loading the new one:



OK to save current file? [Y]es [N]o-Abandon File

2. If you answer by pressing “Y”, VEDIT will save the current file with all modifications. If you answer by pressing “N”, VEDIT will forget any changes that have been made since the last time file was saved. Once the new file is loaded there is no turning back. If you press “N” and answer the next prompt, those changes are lost!
3. Next, VEDIT will prompt for the name of file you would now like to edit using its standard filename dialog box. Either enter the filename or immediately press <Enter> for “Point & Shoot” file selection.

If VEDIT does not find a file with the name you enter, it will create the file. If the file does exist, VEDIT will load the file and place the cursor at the beginning of the file.

Notes: VEDIT knows if a file has been modified since it was loaded or last saved. It uses this information to skip confirmation prompts when you abandon unmodified files and does not re-write unmodified files to disk.

To edit a file in a particular edit buffer (instead of the next available buffer), use **{FILE, Buffer Switch}** (<F4>) to switch to the desired buffer. Then select **{FILE, New File}**.

If you start VEDIT without a file name (or enter another edit buffer) and begin typing, you will end up with text but no associated filename. Since there is no filename, VEDIT can’t know where to save the text. Use **{FILE, Save As}** to select a filename and save the text to disk.

See Also: **{FILE, Open File (new buffer)}**, **{FILE, Buffer Switch}**, **{FILE, Insert File}**

Open (new buffer)

Select a file for editing in new buffer.

Keystroke Equivalent:	<Alt>-<O>, This is a Keystroke Macro that selects “Open File” without creating a new window. <Alt>-<Y>, This is a Keystroke Macro that selects “Open File” and creates a horizontal window.
Description:	With VEDIT, you may edit one or more files at a time. Each file is edited in one of the 37 “edit buffers”. Use “Open File” when you want to simultaneously edit another file in the next available edit buffer. “Open File (new buffer)” automatically selects unused buffers for the additional files you want to simultaneously edit. Normally, the next file loaded will go into buffer “A”, then “B” and so on up to “Y”. However, if one of these buffers is already in use (e.g as a Text Register) it is skipped and the next available buffer selected. To simultaneously edit another file: 1. Select {FILE, Open File (new buffer)} or press <Alt>-<O> or <Alt>-<Y>. 2. VEDIT will prompt for the name of file you would like to edit using its standard filename dialog box. Either enter the filename or immediately press <Enter> for “Point & Shoot” file selection. If VEDIT does not find a file with the name you enter, it will create the file. If the file does exist, VEDIT will load the file and place the cursor at the beginning of the file. If <Alt>-<O> was used, you will now be editing the new file in the same window as before. If no windows are displayed, the entire screen will be used for the new file. If <Alt>-<Y> was used, you will now be editing the new file in a new horizontal window. If {FILE, Open File} was used, VEDIT will prompt whether you want to open a new window too: Open New Window too (Y/N)?

Press “Y” if you want to view two or more files simultaneously.
Press “N” if you prefer to use the full screen for the current file you are editing; this is the typical case unless you need to visually compare the files.

If you answer “Y”, VEDIT will give you an additional prompt:

Vertical/Horizontal Window (V/H)?

Pressing “V” displays the files side by side, while “H” displays the files on top to bottom.

Notes: VEDIT requires enough disk space to accommodate approximately twice the actual file size. You should always work with at least that amount of disk space available.

If you have split the screen into windows, you can “zoom” the current window to full screen by selecting **{WINDOW, Zoom}**.

See Also: “Edit Buffers and Windows” in Chapter 3 (Editing Guide).
{FILE, New File (same buffer)}, **{FILE, Buffer Switch}**
{WINDOW, Create} **{WINDOW, Switch}**, **{WINDOW, Zoom}**

Buffer Switch

Change current edit buffer.

Keystroke Equivalent: <F4>, This is a Keystroke Macro.
Alt-<T>, This is a Keystroke Macro that toggles to the next edit buffer in use.

Description: When simultaneously editing multiple files, the easiest way to switch to another file is with <Alt>-<T>. This Keystroke Macro toggles to the next edit buffer in use. If necessary, press <Alt>-<T> repeatedly until you reach the desired file.

You can also switch directly to a given file by switching to its corresponding “edit buffer” with **{FILE, Buffer Switch}**. Of course, you will need to know the name of the edit buffer. If you simply use <Alt>-<T>, there is no need to know the names of the edit buffers.

To switch to a desired edit buffer:

1. Select **{FILE, Buffer Switch}** (<F4>). VEDIT will prompt you for the buffer name:

Enter Edit Buffer Name:

2. Enter the buffer's name, **A** through **z** (upper or lower case), **0** though **9** or **@**. VEDIT's "main" edit buffer is named **@**.

Entering any invalid character or simply pressing <Enter> will also select the **@** buffer.

Whenever an edit buffer other than **@** is selected (or "active"), the buffer's name is displayed in the left hand corner of the status line. For example, when buffer **A** is selected, "**#A**" is displayed on the status line.

Notes: Selecting **{HELP, Edit Buffers}** displays the name of each edit buffer in use, the name of the attached window(s) and the associated filename, if any. This display is useful when you forget precisely what files you are currently editing.

Edit buffers are a special form of VEDIT's Text Registers. Switching to an edit buffer converts the underlying Text Register into an edit buffer. Any existing Text Register contents are retained in the edit buffer.

The main difference between simple Text Registers and edit buffers is that edit buffers can have associated filenames while simple Text Registers cannot. The only way to convert an edit buffer back into a simple Text Register is with **{FILE, Exit}** or **{FILE, Quit}**; this results in an empty Text Register.

You can insert the contents of an edit buffer using **{BLOCK, Insert Register}** or **T-REG INSERT**.

However, you cannot copy or move a block of text to an edit buffer; this prevents you from accidentally clobbering another file you are editing.

See Also: **{BLOCK, Insert Register}**
{FILE, Open File}, **{FILE, Exit}**, **{FILE, Quit}**
{HELP, Edit Buffers}

Insert File

Insert an entire file at the cursor position.

Keystroke Equivalent:

(none)

Description:

“Insert File” inserts an entire file at the current cursor position. This can be helpful for piecing a file together from various text fragments, or for starting a newly created file with contents of a “template” file.

To insert the contents of another file:

1. Place the cursor at the location where the file should be inserted; the file will be inserted just before the cursor.
2. Select **{FILE, Insert File}**.
3. VEDIT will prompt for the desired filename using its standard filename dialog box. Either enter the filename or immediately press <Enter> for “Point & Shoot” file selection. The cursor remains on the same character which now follows the inserted file.

Notes:

There are no limits on the size of file which can be inserted. This is why you should use **{BLOCK, Write to Disk}** and **{FILE, Insert File}** when copying or moving very large amounts of text between files.

See Also:

{BLOCK, Write to Disk}

Directory

Display directory of files and/or change default directory.

Keystroke Equivalent:

(none)

Description:

“Directory” displays the contents of a disk directory using VEDIT’s “Point and Shoot” file selection. Selecting a file has no effect, it simply returns to your editing. However, VEDIT remembers if you switch to another directory using the “Point & Shoot” facility and makes the last directory you select the new “default directory”.

The “default directory” is the directory VEDIT uses for file selection when you don’t explicitly specify a path.

To display a directory of files:

1. Select **{FILE, Directory}**.

VEDIT will prompt for the desired filename using its standard filename dialog box.

2. Enter the initial “filter” to be used for displaying the desired files. For example, to see all `.txt` files enter `*.txt`. The desired files will be displayed using “Point & Shoot”.
3. When done viewing the files, either press <Esc> or select any arbitrary file.

To switch to another directory and make it the “default” directory:

1. Select **{FILE, Directory}**.
2. At the filename prompt press <Enter> to enter VEDIT’s “Point & Shoot” file selection. Use the normal technique to switch to another directory.
3. Press <Esc> to exit and return to normal editing.

See also: “Point and Shoot File Selection” in Chapter 3 (Editing Guide).

Save As

Name a file and save contents of buffer.

Keystroke Equivalent: (none)

Description: “Save As” lets you save the file you are currently editing under a new name. This function immediately saves the current file contents under the new name. You will then see the new name appear on the status line.

“Save As” also lets you save the contents of an edit buffer which currently has no filename associated with it.

To save the current file under a new name:

1. Select **{FILE, Save As}**.
2. VEDIT will prompt for the desired filename using its standard filename dialog box. Either enter the filename or immediately press <Enter> for “Point & Shoot” file selection.

Notes: If you start VEDIT without a filename and enter some text, you can not save that text to disk until you give it a filename. If you try to save it using **{FILE, Exit}**, you will get the error message “NO OUTPUT FILE” and your text will not be saved to disk. To save this text simply select **{FILE, Save As}** and specify the desired filename; you can then exit VEDIT with **{FILE, Exit}**.

There is no need to select **{FILE, Save and Continue}** immediately after **{FILE, Save As}**; the latter function saves the text itself.

See Also: **{FILE, Save and Continue}**, **{FILE, Exit}**,

Save and Continue

Save the current file and continue editing.

Keystroke Equivalent: (none)

Description: The **{FILE, Save and Continue}** item is your safeguard against disaster! It saves any changes you have made so they can not be lost by power failure, or most major mistakes. You should use this feature frequently since it is important and doesn't take much time.

To save your file with all current changes and continue editing:

1. Select **{FILE, Save and Continue}**.

If you have made any changes since the last time you saved the file, the message “Please wait for disk” will appear on status line while VEDIT is saving your changes. If you did not make any changes, VEDIT will know that and does not bother to save file again.

2. Continue editing the file. Remember to regularly save your changes.

Notes: If you start VEDIT without a filename and enter some text, you can not save that text to disk until you give it a filename. If you try to save it using **{FILE, Save and Continue}**, you will get the error message “NO OUTPUT FILE” and your text will not be saved to disk. To save this text select **{FILE, Save As}** and specify a filename; you can then continue editing.

See Also: **{FILE, Save As}**, **{FILE, Exit}**

Exit (Save)

Save the file or files being edited. This is the normal way to exit VEDIT.

Keystroke Equivalent: <Alt>-<X>, This is a Keystroke Macro

Description: When only one file is being edited, this item simply saves the file to disk and exits VEDIT. When two or more files are being edited, you are prompted with:

```
Exit (save) this file? [Y]es [N]o; Exit [A]ll; [C]ancel
```

[Y]es Saves the file being edited in the current edit buffer and switches to the next edit buffer in use. Other files are not affected. (The edit buffer, other than @, is converted back into an empty Text Register).

[N]o Cancels the operation; same as [C]ancel or pressing <Esc>.

Exit [A]ll Saves all files currently being edited and exits VEDIT.

[C]ancel Cancels the operation; same as [N]o or pressing <Esc>.

Notes: Selecting “Exit All” only saves text in those edit buffers which have filenames assigned. However, as a safety precaution, it does not exit VEDIT if the main @ buffer has no assigned filename.

The bottom of the screen displays the name of the file being saved following the message “Saving...” or “File not altered...”. The later message is used when VEDIT knows that no alterations were made since the file was last saved to disk.

{FILE, Exit} is also found in the Escape Menu.

See the “Notes” for {FILE, Save As} if you encounter the error message “NO OUTPUT FILE”.

See Also: {FILE, Save As}, {FILE, Save and Continue}, {FILE, Quit}

Quit (Abandon)

Quit the file or files being edited and exit VEDIT. This abandons any changes that have been made.

Keystroke Equivalent: (none)

Description: When only one file is being edited and it has not been altered, this item simply exits VEDIT. If the file has been altered, you are prompted with:

```
Abandon altered file? [Y]es [N]o
```

[Y]es Abandons any changes made to the file and exits VEDIT.

[N]o Cancels the operation; same as pressing <Esc>.

When two or more files are being edited, you are prompted with:

```
Quit (abandon) this file? [Y]es [N]o; Quit [A]ll; [C]ancel
```

[Y]es Abandons the file in the current edit buffer and switches to the next edit buffer in use. Other files are not affected. (The edit buffer, other than @, is converted back into an empty Text Register).

[N]o Cancels the operation; same as [C]ancel or pressing <Esc>.

Quit [A]ll Abandons all files currently being edited and exits VEDIT.

[C]ancel Cancels the operation; same as [N]o or pressing <Esc>.

Notes: Abandoning a file causes VEDIT to “forget” any changes you have made since the last time the file was saved. Needless to say, use this capability with caution.

Selecting **{FILE, Quit}** while editing multiple files with multiple windows also deletes the window(s) associated with the current edit buffer.

{FILE, Quit} is also found in the Escape Menu.

See Also: **{FILE, Exit}**

Goto Menu

Keystroke Equivalent: <Alt>-<G>, This is a Keystroke Macro.

Introduction: The Goto Menu provides quick access to a wide variety of locations in your file. In addition to items in this menu, VEDIT offers numerous Edit Functions for moving about your text. If you do not find an expected “goto” type operation in this menu, look in Chapter 6 (Edit Function Reference).



Beginning of File

Position the cursor at the absolute beginning of the file.

Keystroke Equivalent: <Alt>-<A>, This is a Keystroke Macro.

Description: To move the cursor to absolute beginning of the file:

1. Select {**GOTO, Beginning of File**} (<Alt>-<A>). In a very large file, this may take a few moments. When completed, the status line will read “Line 1” and “Column 1”.

Notes: In small files, {**GOTO, Beginning of File**} has the same effect as {**GOTO, Home in Memory**}.

When you’re working in a large file or have limited memory, VEDIT may have parts of the file on disk in a temporary file. Unlike {**GOTO, Home in Memory**} which only moves to the beginning of what is in memory, {**GOTO, Beginning of File**} causes VEDIT to read back through this disk file to find the absolute beginning of your text.

Refer to the description of {**GOTO, Home in Memory**} for a more complete discussion of how VEDIT manages large files.

See Also: **{GOTO, Home in Memory}**
Appendix C (Memory Management)

Home in Memory

Position the cursor at the beginning of the text which is currently in memory.

Keystroke Equivalent: (none)

Description: VEDIT manages small files by keeping the entire file in high speed “RAM” computer memory. Unfortunately, larger files can be too big to entirely fit within the RAM space available. To solve this problem, VEDIT keeps only part of the file near the cursor location in memory and the rest in a temporary file on your disk drive.

As you move the cursor through a large file, VEDIT occasionally “reads in” parts of the file and “writes out” other parts to keep text near the cursor in memory.

{GOTO, Home in Memory} moves the cursor to the beginning of what is currently in memory without “reading in” or “writing out” any text; it therefore acts instantaneously.

If the cursor is already at the beginning of the text that is in memory, **{GOTO, Home in Memory}** causes VEDIT to “read in” and “write out” another section of text toward the beginning of the file and again move the cursor to the beginning of that text.

Using **{GOTO, Home in Memory}** and **{GOTO, Zend in Memory}** are useful for quickly moving backward or forward through files in fairly large jumps.

To position the cursor at the beginning of the text which is currently in memory:

1. Select **{GOTO, Home in Memory}**.

To then move the cursor further toward the beginning of the file, repeatedly select **{GOTO, Home in Memory}**.

Notes: When you are editing a file small enough to entirely fit in memory, **{GOTO, Home in Memory}** has the same effect as **{GOTO, Beginning of File}**.

VEDIT's "read in" and "write out" operations are referred to as "Auto-Buffering". Auto-buffering attempts to read and write entire lines of text; however with lines longer than 8,000 bytes, this is not always possible.

Refer to the word "LINE" on the status line. If the "L" is capitalized, the beginning of the file is in memory, otherwise the beginning of the file has been written to disk.

See Also: **{GOTO, Beginning of File}**, **{GOTO, Zend in Memory}**,
{GOTO, End of File}

Zend in Memory

Position the cursor at the end of the text which is currently in memory.

Keystroke Equivalent: (none)

Description: **{GOTO, Zend in Memory}** moves the cursor to the end of what is currently in memory without "reading in" or "writing out" any text; it therefore acts instantaneously.

To position the cursor at the end of the text which is currently in memory:

1. Select **{GOTO, Zend in Memory}**.

To then move the cursor further toward the end of the file, repeatedly select **{GOTO, Zend in Memory}**.

Notes: Refer to description of **{GOTO, Home in Memory}** for a more complete discussion of how VEDIT manages large files.

When you are editing a file small enough to entirely fit in memory, **{GOTO, Zend in Memory}** has the same effect as **{GOTO, End of File}**.

The word "Zend" is derived from the original 1980 VEDIT, you can also think of it as as "Zee End of Zee Memory"!

Refer to the word "LINE" on the status line. If the "INE" is capitalized, the end of the file is in memory, otherwise it is on disk.

See Also: **{GOTO, Beginning of File}**, **{GOTO, Home in Memory}**,
{GOTO, End of File}

End of File

Position the cursor at the absolute end of the file.

Keystroke Equivalent: <Alt>-<Z>, This is a Keystroke Macro.

Description: To move the cursor to absolute end of the file:

1. Select **{GOTO, End of File}** (<Alt>-<Z>). In a very large file, or with a heavily loaded file system, this may take a few moments and you will see the message “Please wait for disk”.

Notes: In small files, **{GOTO, End of File}** has the same effect as **{GOTO, Zend in Memory}**.

Refer to description of “Home in Memory” for a discussion of how VEDIT manages large files.

See Also: **{GOTO, Beginning of File}**, **{GOTO, Home in Memory}**, **{GOTO, Zend in Memory}**

Line

Move the cursor to the beginning of a specific line.

Keystroke Equivalent: (none)

Description: Lines are numbered starting from the first line in your file. The current line number is indicated on the Status Line. There are no actual line numbers as you might find in a BASIC language program and the line numbers do not “stick” to real text lines. If you insert or delete a line in the middle of your file, line numbers of all lines which follow will be changed.

{GOTO, Line #} is useful when working with large files. If you remember an exact or even approximate line number of a particular location in your file, you can get to it very quickly.

To go to a specific line in your file:

1. Select **{GOTO, Line #}**.
2. VEDIT will prompt you for the line number. Enter the desired line number and press <Enter>.

You can also enter numeric expressions such as “(1245+858)/2”.

Notes: The mouse can also be used to quickly access relative positions in the file.

See Also: {GOTO, Set Text Marker}, {GOTO, Goto Text Marker}

Column #

Move the cursor to a specific column in the current line.

Keystroke Equivalent: (none)

Description: To go to a specific column in the current line:

1. Select {GOTO, Column #}.
2. VEDIT will prompt you for the column number. Enter the desired column number and press <Enter>.

You can also enter numeric expressions such as “1980/4”.

Notes: This item is primarily useful when editing very long lines, i.e. longer than 100 or 200 characters. For example, if you know that a line is about 2000 characters long and you want to go to the middle of it, you could simply go to column 1000. This obviously is faster than pressing **SCROLL RIGHT** fifty times.

Set Text Marker

Set a Text Marker at the current cursor location.

Keystroke Equivalent: <Ctrl>-<S>, This is a Keystroke Macro.

Description: Ten invisible “Text Markers” are available for remembering locations in your file and returning you to them later. These markers “stick” to the character at which you set them. If you insert or delete text in front of a Text Marker, the marker will move along with the text to which it was attached.

Marker information is not saved in files. As a result, markers are maintained only until you begin editing another file or exit VEDIT.

To set a Text Marker:

1. Move the cursor to the desired position.
2. Select {GOTO, Set Text Marker}.

VEDIT will prompt you with:

Marker Number: (0-9)

3. Press the desired marker number **0** through **9**. As a shortcut you can press <Ctrl>-<S> again (or <Enter> or any function key) to select marker **0**.

Notes: Text Markers are invisible; you can't see where they are aside from using **{GOTO, Goto Text Marker}** to position the cursor there.

While there is no limit on actual file size, Text Markers cannot span more than 8,338,607 characters.

You can think of Block Markers as special purpose Text Markers; but don't confuse them.

{BLOCK, Remove Markers} only removes the block markers, it does not affect the Text Markers. There is no way to remove or clear Text Markers once they are set.

Certain editing operations affect both Text and Block Markers:

- Any operation which deletes text at or around a marker leaves the marker set at the first character past the deleted text. "Undo"ing deletion will not restore the marker to its original position.
- Moving a block which contains a marker does not move the marker. The marker behaves exactly as if the original text was deleted.
- Reformatting a paragraph moves any markers located within that paragraph to the last character in the paragraph.

See Also: **{GOTO, Goto Text Marker}**

Goto Text Marker

Return the cursor to a previously set Text Marker.

Keystroke Equivalent: <Ctrl>-<G>, This is a Keystroke Macro.

Description: "Goto Marker" moves the cursor to the location of a desired Text Marker. The marker must first be set using **{GOTO, Set Text Marker}**.

You can have up to 10 markers set at any one time.

To move back to a previously marked position:

1. Select **{GOTO, Set Text Marker}**.
2. VEDIT will prompt you for the Text Marker. Press the desired marker number “0” through “9”. As a shortcut you can press <Ctrl>-<S> again (or <Enter> or any function key) to select marker “0”.

Notes: See the description of **{GOTO, Set Text Marker}** for a discussion of Text Markers.

When working with a large file, **{GOTO, Goto Text Marker}** may take a few moments to execute.

See Also: **{GOTO, Set Text Marker}**

Matching () {} []



Keystroke Equivalent:

<Alt>-<F3>, This is a Keystroke Macro.

Description:

This item examines the character at the current cursor position and attempts to locate a logically matching character. The characters which have logical matches are “(”, “)”, “{”, “}”, “[”, “]”, “<” and “>”. These characters are frequently used in program source code, arithmetic expressions and other logical constructs.

{GOTO, Matching ()} is designed to help you move from one end of these logical units to the other and to locate mismatches.

For example, if you place the cursor at the first “(” in following expression and select this item, VEDIT will move the cursor to the last “)” instead of the first one it finds.

((a/1) + (b/2))

While it is easy to see that all parentheses properly match in the above expression, some expressions and program source code routines and functions can be so lengthy and complex that it may not be obvious.

{GOTO, Matching ()} can check for proper matching over thousands of lines of source code.

VEDIT matches these characters in either direction. If the cursor is on an opening character—“(”, “”, “{”, “[” or “<”—VEDIT searches forward for a match. If the cursor is on a closing character—“)”, “}”, “]” or “>”—VEDIT searches backward for a match.

To locate a matching character:

1. Place the cursor on character you would like to match. If the cursor is not located on such a character when this item is selected, the first of these eight characters will be found for you.
2. Select **{GOTO, Matching ()}**.

The cursor will be moved to the matching character if there is one. Otherwise an error message is given.

HINT:



Here's a trick for checking an entire file for proper character matching. Place a “(” at the beginning of the file and a “)” at the end. Place the cursor at one of these two characters and select **{GOTO, Matching ()}**. If the cursor moves all the way to other end of the file and doesn't display an error message, everything is properly paired.

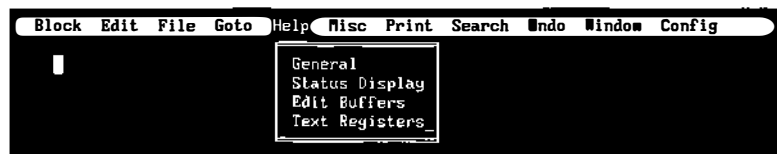
Help Menu

Keystroke Equivalent:

<Alt>-<H>, This is a Keystroke Macro.

Introduction:

The Help Menu provides direct access to a wide variety of helpful information. This includes information provided in the online help files and important information related to the current status of VEDIT.



General

Activate VEDIT's online help system.

Keystroke Equivalent: <Alt>-<F1> while no menu or prompt is displayed; this is the **HELP** function.

Description: VEDIT provides an extensive online help system. Using **{HELP, General}** you can quickly locate specific information on nearly every aspect of using VEDIT:

1. Editing functions and equivalent keystrokes are listed.
2. Built-in Keystroke Macros supplied with VEDIT are itemized.
3. Box drawing characters assigned to equivalent keystrokes are displayed.
4. Help on specific keyword topics can be requested.

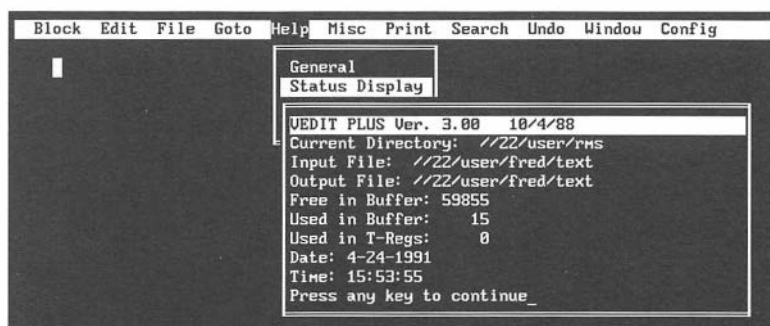
See Also: "Requesting Help" in Chapter 3 (Editing Guide).

Status Display

Display VEDIT's version, release date and status information.

Keystroke Equivalent: (none)

Description: The VEDIT Status Display provides important information about the copy of VEDIT you are using and its present state. It is displays as in the following pop-up window:



The information which
will be displayed is:

Product name version and release date.	You may need this information for technical support or upgrade assistance. It describes exactly which product you are using, the revision number and the date this revision was first released.
Current directory	This is the default directory for all file operations. It can be changed using {FILE, Directory} .
Input File	The file VEDIT last loaded for you to edit. If you started VEDIT with a single filename, this will be the same as the “Output File”.
Output File	The file to which VEDIT will save your editing changes. If you started VEDIT with a single filename, this will be the same as “Input File”.
Free in Buffer	This is the RAM (memory) space, in characters, which remains available for holding your file in the current edit buffer. If this value drops too low, VEDIT will “write out” some of the file to a temporary disk file and keep this value somewhat constant.
Used in Buffer	This is RAM (memory) space, in characters, which is filled with your file in the current edit buffer. If you are editing a large file, this may be less than actual size of your file.
Used in T-Regs	“T-Regs” refers to “Text Registers”. This is number of characters which are currently stored in all Text Registers. The maximum value is about 55,000 characters, but may be less if memory is limited. The value shown does not include memory used for edit buffers.
Date	The current date as reported by QNX.
Time	The current time as reported by QNX.

If either date or time is incorrect, you can use the QNX **date** command to set it. In some cases the **rtc** command may also be relevant.

Press any key when you are done examining the above information.

Edit Buffers

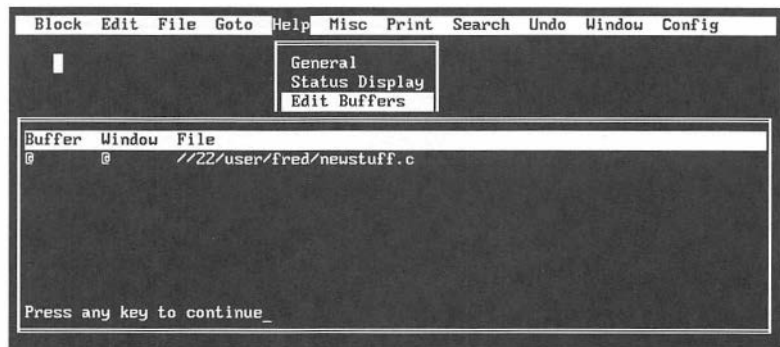
Displays names of the current edit buffers, which window they are in and the names of the files being edited.

Keystroke Equivalent:

(none)

Description:

The names of all edit buffers currently in use are listed with corresponding filenames and attached windows. The full pathname is displayed. Some edit buffers don't have any associated filename.



In the above example, there are four edit buffers in use. Buffer "A" is displayed in window "A", while buffers "B", "C" and @ (the main buffer) alternate being displayed in the main window @.

Notes:

The main Window for VEDIT is designated by @. If other Windows have been specified, they will be listed by their designated name opposite their corresponding buffer. A blank in the Window column means that no switch to designated buffer has yet been made.

See Also:

{FILE, Buffer Switch}, {FILE, Open File (new buffer)}

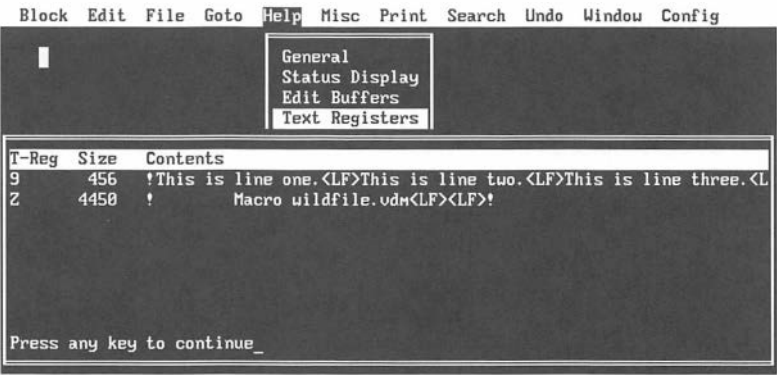
Text Registers

Display active Text Registers: name, size and some of their contents.

Keystroke Equivalent:

(none)

Description: {**HELP, Text Registers**} displays the names of non-empty Text Registers, their size and up to the first 60 bytes of their contents. An example is:



Notes: If you copy or move text into a Text Register (e.g with {**BLOCK, Copy to Register**} or **T-REG MOVE**), the register will show up in this help window. However, if you then convert the Text Register into an edit buffer using {**FILE, Buffer Switch**}, the register will no longer display here because it is now an edit buffer.

See Also: {**HELP, Edit Buffers**}, {**FILE, Buffer Switch**}

Misc Menu

Keystroke Equivalent: <Alt>-<M>, This is a Keystroke Macro.

Introduction: The Misc Menu is an assortment of items which are not closely related to other Main Menu topics. The {**Keyboard Layout**} submenu is also found here.



ASCII Table

Display table of ASCII character values and optionally enter a character by ASCII value.

Keystroke Equivalent:

(None)

Description:

“ASCII Table” displays a table of all possible characters with their corresponding decimal value. This includes all control, alphanumeric and graphic characters. The display is three “windows” long and at any time you can insert any particular character into your text.

The ASCII table is particularly useful for entering graphics characters where you need to first see the characters before deciding which to enter.

To display the ASCII table and insert a desired character:

1. Select {**MISC, ASCII Table**}.

If necessary, press <Enter> once or twice until the desired character is displayed.

2. Press “I” if you want to insert a particular character. VEDIT will prompt you with:

Enter Decimal Value to Insert:

3. Type the ASCII value displayed in the table for the particular character followed by <Enter>.

Notes:

“ASCII” stands for “American Standard Code for Information Interchange”. Each character of your text is represented inside of the computer by a standard numeric value. When you type a character on the keyboard, it is converted to a corresponding numeric value for storage in memory and on disk. Certain characters do not have corresponding keys on the keyboard and therefore can’t be directly typed. On PC compatible computers, these include most of the line drawing and graphic characters.

If you know the hexadecimal value of a character you can enter it by pressing and releasing the <Alt> key then typing the two-digit hex value.

There will appear to be no character display for values 00 and 255 because when displayed, these characters are indistinguishable from a space.

VEDIT can display control and graphics characters in several ways. This is described under “Displaying Control and Graphics character” in Chapter 3 (Editing Guide).

You can also insert control characters using **ENTER CTRL**.

See Also: “Entering Control and Graphics Characters” in Chapter 3 (Editing Guide).

ENTER CTRL

Shell

Suspend VEDIT and display the QNX prompt.

Keystroke Equivalent: (none)

Description: “Shell” temporarily suspends operation of VEDIT and “shells out” to QNX. You can then run any QNX command and/or programs. These commands can range from simple directory and file management commands (e.g. **ls**, **cp**, **wc**) to running other application programs. This lets you temporarily interrupt your current editing to perform other tasks.

{**MISC**, **Shell**} does not affect your files, it merely suspends VEDIT. When you return to VEDIT, you will find everything exactly as it was when you left it.

To suspend VEDIT and bring up a shell:

1. Select {**MISC**, **Shell**}.

You should now see the QNX command prompt.

2. You can now execute QNX commands and other software.
3. When you are ready to return to VEDIT, hit <Ctrl>-<D> to terminate the shell. VEDIT may ask you to press any key before it restores the screen.

WARNING:

If you forget to return to VEDIT before you turn off or reboot your computer, any changes made since the last time your files were saved will be lost! Therefore, it is a good habit to save all file changes before shelling out.



DO NOT delete files whose names begin with the name of a file you are editing and end with a “\$” sign. These may be temporary files in use by VEDIT; if you delete or modify them, you will destroy or corrupt your file(s)!

The attempt to shell out of VEDIT may fail if there is not enough free memory to start up a new shell. Usually this problem can be solved by closing some files to reduce the buffer memory VEDIT is using and trying again.

{**MISC, Shell**} will not work unless the **\$SHELL** environment parameter is properly set. This is usually set automatically from your password entry at login time.

See Also:

{**MISC, Run QNX Program**}

Run Program

Execute a QNX command or other program and return to VEDIT.

Keystroke Equivalent:

(none)

Description:

{**MISC, Run QNX Program**} lets you execute most QNX commands or application programs. Unlike {**MISC, Shell**}, {**MISC, Run QNX Program**} automatically returns to VEDIT as soon as the command or program is completed. Simply put, use {**MISC, Shell**} for multiple commands or programs and use {**MISC, Run QNX Program**} for a single command or program.

To execute a single QNX command or other program:

1. Select {**MISC, Run QNX Program**}.

VEDIT will give you the following prompt:

Enter QNX program / command:

2. Enter the desired QNX command or program name. You may enter any valid command, execute any program which may be run using a single command. Ample space is provided to enter most types of commands, including those which require several arguments.

WARNING:



See the warnings for **{MISC, Shell}**.

Notes: All Notes for **{MISC, Shell}** also apply here.

If you frequently run a program which requires a series of commands to execute, you should consider creating a “shellscript” file and using **{MISC, Run QNX Program}** to execute it. Refer to your QNX manual for information on “shellscripts”.

See Also: **{MISC, Shell}**

Color Toggle

Select the next suggested color combination.

Keystroke Equivalent: **<Alt>-<R>**, This is a Keystroke Macro.

Description: Built into VEDIT are about 15 suggested color combinations for displaying text on the screen.

{MISC, Color Toggle} steps VEDIT to the next suggested color in the current window. As supplied, VEDIT uses a default color combination. Using this item, you can select others “on the fly” to provide some variety as you work.

To change the current color combination and save it:

1. Ensure that only one window is on the screen. If necessary select **{WINDOW, Remove All}**.
2. Repeatedly select **{MISC, Color Toggle}** (**<Alt>-<R>**) until the text appears in the desired color.
3. Select **{CONFIG, Save to disk}** if you want to make the new color combination permanent.

Notes: Color combinations selected with “Color Toggle” item are not the only combinations possible. Editing the VEDIT configuration files (see Chapter 7) allows other changes.

Color monitors connected to color display adapter cards are capable of displaying each of these combinations correctly. Monochrome monitors or black and white monitors connected to color adapter cards will not.

See Also: “Change Screen Display Parameters” in Configuration, Chapter 7.

Load/Execute Macro

Loads and executes a Command Macro.

Keystroke Equivalent: (none)

Description: This item lets you load and immediately execute a VEDIT “Command Mode” macro in one quick step. Examples include **wildfile.vdm**, **print.vdm** and any of the **.vdm** files from the supplied macro libraries.

To load and execute a (Command Mode) macro:

1. Select {**MISC, Load/Execute Macro**}.
2. VEDIT will prompt you for the the name of the Text Register in which to load the macro. Typically used names are **1** through **9** and **G** through **Z**. Type the appropriate letter/digit or simply press <Enter> to use the default register **Z**.
3. VEDIT will prompt for the desired filename using its standard filename dialog box. Either enter the filename or immediately press <Enter> for “Point & Shoot” file selection. Since these macros are usually stored as **.vdm** files in the “VEDIT Home Directory”, e.g. **/usr/lib/vedit**, the “Point & Shoot” “filter” would in this case default to **/usr/lib/vedit/*.vdm** for very quick macro selection.

Notes: When entering the filename, you can leave off the default **.vdm** extension; you can also leave off the pathname to the “VEDIT Home Directory”.

VEDIT’s “Point and Shoot” file selection makes it very easy to load and execute macros. Typically you only have to select **{MISC, Load/ Execute Macro}**, press <Enter> twice in response to the prompts and then move the cursor (or mouse) to the desired macro name; only the **.vdm** files in the “VEDIT Home Directory” will be displayed.

Don’t confuse these “Command Macros” stored in files and Text Registers with “Keystroke Macros”. Command Macros are programs written in the VEDIT macro language. Keystroke Macros are “strings” of key-presses, possibly including a few macro language commands, which are assigned to “hot-keys”.

Information from **{HELP, Text Registers}** is useful to determine which registers are empty and therefore available for holding Command Macros.

The Text Register selected for loading cannot be the current edit buffer nor any other edit buffer.

See Also: “Point and Shoot” in Chapter 3 (Editing Guide).
Command Macros (Chapter 4).
{HELP, Text Registers}, {MISC, Execute Macro}

Execute Macro

Executes the Command Macro stored in a Text Register.

Keystroke Equivalent: (none)

Description: Once a macro has been loaded into a register, it may be executed using **{MISC, Execute Macro}**.

Except for the operation of loading a macro into a register, this item is identical to **{MISC, Load/Execute Macro}**.

{MISC, Execute Macro} is often to used to repeatedly execute a macro after it is loaded and executed for the first time using **{MISC, Load/ Execute Macro}**.

Notes: You can execute a Command Macro stored in an edit buffer, *but not in the current edit buffer*. To execute a macro in the current edit buffer, you must first switch to another edit buffer using **{FILE, Buffer Switch}** (<F4>).

See Also: **{MISC, Load/Execute Macro}**

**Keyboard Layout
(Submenu)**

Select the “Keyboard Layout” submenu.

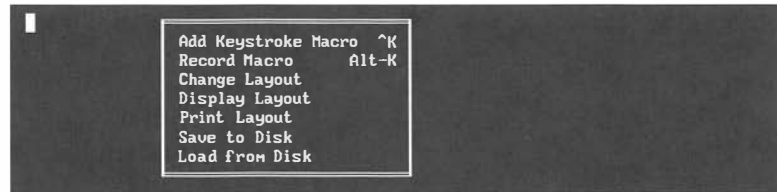
Keystroke Equivalent: (none)

Description: VEDIT’s keyboard layout is completely configurable. Each basic Edit Function can be assigned to any desired key or keys. You can also add as many Keystroke Macros and “hot-keys” as desired to the layout to speed your editing.

This submenu lets you change the basic layout, add new Keystroke Macros and display the current layout. You can also save the layout to a disk file and load a new layout at any time.

Any changes you make to the layout are temporary unless you make them permanent by selecting **{CONFIG, Save to disk}**.

Selecting {**MISC, Keyboard Layout**} displays the following submenu:



Each item of the submenu is explained below.

Notes: Keystroke Macros are introduced in Chapter 3 (Editing Guide).

VEDIT's default keyboard layout includes many pre-defined Keystroke Macros for selecting menu items; these are often called "hot-keys". VEDIT's menus display all available "hot-keys".

The key to which a Keystroke Macro is assigned can actually be a sequence of up to 16 keys (often called an "Escape Sequence"). This lets you define more Keystroke Macros than there are function and control keys. For example, you might use <gray -> as the "lead-in" to an entire group of Keystroke Macros. Pressing <gray -> and then **A** could play back one Keystroke Macro, pressing <gray -> and then **B** could play back another, and so on.

Keystroke Macros may consist of just a single keystroke such as **CURSOR UP**.

This makes it easy to have alternate keys which perform the same basic editing function.

Technically, VEDIT does not distinguish between the basic keyboard layout (keys assigned to the Edit Functions) and Keystroke Macros; VEDIT's internal "keyboard layout table" simply lists the key or keys you press and the editing function or function(s) to be performed. For the user's benefit, key(s) which perform a single Edit Function are considered part of the basic keyboard layout displayed by **HELP** and {**HELP, General**}; everything else is considered a Keystroke Macro.

Add Macro

Define a new Keystroke Macro or change an existing one.

Keystroke Equivalent:

<Ctrl>-<K>, This is a Keystroke Macro.

Description:

This item lets you add (define) a new Keystroke Macro or change an existing one. Compared to **{MISC, Keyboard Layout, Record Macro}**, this item has the advantage of letting you edit the macro as you enter it. It should also be used for macros which access the VEDIT macro language.

To add (define) a new Keystroke Macro:

1. Select **{MISC, Keyboard Layout, Add Keystroke Macro}**.

You will see the following prompt:

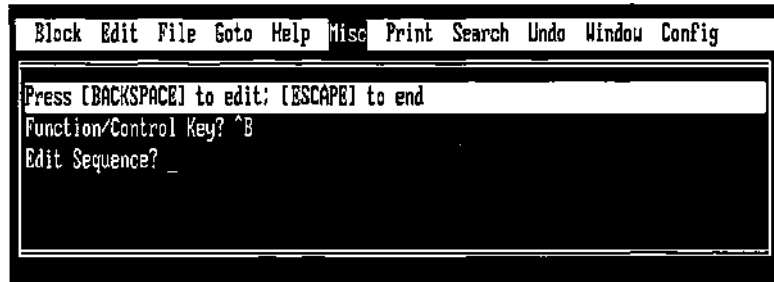
Press <Backspace> to edit; <Esc> to enter
Function/Control Key?

2. Press the function or control key(s) to which this Keystroke Macro should be assigned. This is the key(s) which you will later press to play back the macro. Typically a Keystroke Macro is assigned to a single keystroke or keystroke combination, e.g. <F11>, <Shift>-<F10> or <Ctrl>-<W>, which is currently unused or whose current assignment you want to overwrite.

You can edit the “Function/Control Keys” sequence using **BACKSPACE**.

3. When “Function/Control Key” is correct press **ESCAPE**.
4. If the “Function/Control Key” you entered is already assigned to a Keystroke Macro or Edit Function, you will be prompted with “Redefine existing key (Y/N)?” for confirmation to overwrite the existing assignment.

For example, if you were assigning a Keystroke Macro to <Ctrl>-, you would now see the following display:



5. For “Edit Sequence?” enter the exact sequence of keystrokes you want the macro to play back each time it is invoked. You can edit the “Edit Sequence” using **BACKSPACE**.
6. When the “Edit Sequence” is correct press **ESCAPE**.

The Keystroke Macro is now defined and ready for use. It will be available until you exit VEDIT (or overwrite it). To make it permanent use {**CONFIG, Save to disk**}.

Modifying an Existing Keystroke Macro

1. Select {**MISC, Keyboard Layout, Add Keystroke Macro**}.
2. Press the “Function/Control Key” assigned to the Keystroke Macro you want to modify. Press **ESCAPE**.
3. In response to “Redefine existing key (Y/N)?”, answer “Y”.
4. For the “Edit Sequence”, press the same function/control key as you pressed in step 2 above. The original macro sequence will be inserted for you. You can then use **BACKSPACE** to erase part of the macro and/or type in additional keystrokes. When finished, press **ESCAPE**.

In effect, VEDIT does not “forget” the Keystroke Macro until you are done defining the new one. This can be helpful, not only for editing existing macros, but also for building a larger Keystroke Macro from other smaller Keystroke Macros.

To erase (delete) a Keystroke Macro (or assignment to an Edit Function):

1. Follow steps 1 - 3 above for “Modifying an Existing Keystroke Macro”.
2. At the “Edit Sequence” prompt immediately press **ESCAPE**.

Notes: To cancel this function you must be at the “Function/Control Key” prompt. Press **BACKSPACE**, if necessary, until all keypresses have been erased, and then press **ESCAPE**.

To use <Esc> or <backspace> as part of the “Function/Control Key” or “Edit Sequence” you must precede that key with **ENTER CTRL** (<Ctrl>-<Q>).

This item uses the Edit Functions **ESCAPE** and **BACKSPACE** which on the console are almost always assigned to the <Esc> and <backspace> keys.

However on CRT terminals the assignments may be different; in all cases the top line in the dialog box displays the exact key(s) to be pressed.

The **REPEAT** function can be used with keystroke Macros to repeat the macro a number of times. A **REPEAT** can also be part of the assigned “Edit Sequence”. The only limitation is that you cannot **REPEAT** a Keystroke Macro which itself contains a **REPEAT**.

See Also: “Keystroke Macros” in Chapter 3 (Editing Guide).
Notes for {**MISC, Keyboard Layout**} above.
{**MISC, Keyboard Layout, Record Macro**},
{**MISC, Keyboard Layout, Display Macros**}

Record Macro

Defines a Keystroke Macro by recording your editing operations while you make them.

Keystroke Equivalent: <Alt>-<K>, This is a Keystroke Macro.

Description: This item lets you record (define) a new Keystroke Macro while you are editing. This makes it easy to record a sequence of editing steps you are going to use over and over again.

This is an alternative to **{MISC, Keyboard Layout, Add Keystroke Macro}** and has the advantage of letting you see the effect of the Keystroke Macro as you enter it. However, “Add Keystroke Macro” has the advantage of letting you edit the macro as you enter it.

You can define most Keystroke Macros using either “Add Keystroke Macro” or “Record Macro” according to your preference. New users will tend to prefer the latter, experienced users the former. However, “Add Keystroke Macro” is required for macros which access the VEDIT macro language.

To record a new Keystroke Macro:

1. Select **{MISC, Keyboard Layout, Record Macro}**.

You will see the following prompt:

Press <Backspace> to edit; <Esc> to end
Function/Control Key?

2. The “Function/Control Key” is entered identically to that for **{MISC, Keyboard Layout, Add Keystroke Macro}** —simply press the desired function or control key(s), use **BACKSPACE** if you make a mistake, and press **ESCAPE** when done.
3. VEDIT is now in “Record Mode” —this is indicated on the status line. Everything you now type becomes part of the Keystroke Macro.
4. To finish recording the new Keystroke Macro you must press the “Stop Record” key which is displayed on the right hand side of the status line.

The “hot-key” for this menu item (i.e. <Alt>-<K>) is the “Stop Record” key. When no “hot-key” is defined, the “Stop Record” key is <Ctrl>-<J> which is available on all keyboards.

See Also:

For the most part, all commentary and notes for **{MISC, Keyboard Layout, Add Keystroke Macro}** apply here too.

Change Layout

Change the key assignment for the basic Edit Functions.

Keystroke Equivalent:

(none)

Description: This item lets you change the key assignments for the basic Edit Functions. You can change just a few assignments or create a completely new keyboard layout after having VEDIT delete all current assignments. VEDIT first gives you the following prompt:

```
Delete existing layout? [E]dit functions only; [A]ll keys; [N]o
```

[E]dit functions only	Deletes the key assignments for all Edit Functions, but leaves all Keystroke Macros intact. This is useful for making substantial changes. By deleting all Edit Function assignments now, you won't be prompted continuously for confirmation to overwrite existing assignments.
[A]ll keys	Deletes the key assignments to all Edit Function and deletes all Keystroke Macros. This is only useful when you want to create a completely new layout.
[N]o	Keeps the current layout so that you can make changes to it.
<Esc>	Press <Esc> to cancel this function now.

VEDIT now lists each Edit Function, one at a time, with its current assignment. When there is no assignment, "not used" is displayed; this will be true for all Edit Functions if you responded to "Delete existing layout?" with "E" or "A". You are then prompted for the new assignment.

You can either enter the new assignment or immediately press <Enter>

to keep the current assignment and advance to the next Edit Function. You can skip forward and backwards over the assignment in order to "edit" them. The following responses are possible:

Function or control key	To change the current assignment simply press the desired function or control key(s). Then press <Enter>.
--------------------------------	---

<Enter>	Immediately press <Enter> to keep the current assignment and advance to the next entry.
-	Press “-” (dash) to remove the current assignment without assigning anything else. This effectively disables the function (unless there is a duplicate key).
=	Permits adding duplicate keys that performs the same Edit Function. VEDIT will list the same Edit Function again; if there is an existing duplicate key, it will be displayed, otherwise “not used” will be displayed. You can then enter the duplicate key assignment or alter and/or delete any existing duplicate key assignment.
B	Returns to the beginning, i.e. to the assignment for BACKSPACE .
E	Ends changing the layout. Any deletions and changes to the current layout have been made. Same as repeatedly pressing <Enter> through the rest of the entries.
P	Goes to the previous entry. Useful when you notice that you have made a mistake and need to back up in order to correct it.
H	Access the online help for this task.

To change a single Edit Function assignment:

1. Select {MISC, Keyboard Layout, Change Layout}.
2. At the “Delete existing layout” prompt, type “N”.
3. At the prompts for each Edit Function repeatedly press <Enter> until you reach the desired function.
4. Change the Edit Function assignment by pressing the new function/control key you want assigned to it. Then press <Enter>.
5. At the prompt for the next Edit Function, simply enter “E” to end this task. You will be returned to normal editing.

6. To make the change permanent, select **{CONFIG, Save to disk}**.



Since it is possible to change the keyboard layout in such a way that you can no longer exit VEDIT, you should save your files before selecting the “Change Layout” function.

Notes:

Some users may prefer to change the keyboard layout by directly editing the “keyboard layout table” file. This is described under **{MISC, Keyboard Layout, Save to Disk}**.

If the new function/control key you enter is already assigned to an Edit function or Keystroke Macro, you are prompted for confirmation to overwrite it.

If VEDIT detects that you have not assigned any key to **ESCAPE**, it automatically assigns it to <Esc>.

This usually ensures that enough Edit Functions are defined for exiting VEDIT—press <Esc> to bring up the **{ESCAPE}** menu; select “Exit File” using <spacebar>; then press <Enter>.

No Edit Function or Keystroke Macro can be assigned to <Ctrl>-<M>, because it is the same as <Enter>.

Similarly, you should not attempt to assign anything to <Ctrl>-<[>, because it is the same as <Esc>.

The following control keys are available for assignment to Edit functions and Keystroke Macros: <Ctrl>-<]>, <Ctrl>-<_>, <Ctrl>-</> and <Ctrl>-<^>.

CRT terminal notes:

This technical description is only applicable to users running VEDIT on externally connected CRT terminals.

The function keys on most CRT terminals send a sequence of characters called an *Escape Sequence*. As the name implies, Escape Sequences typically begin with the Escape character, followed by one to six additional characters. VEDIT fully supports all functions keys including those that send Escape Sequences and 8-bit characters.

Some Escape Sequences end with a Carriage-return character which is identical to <Enter>.

Escape Sequences can also be hand typed, i.e. you first press <Esc> and then another character.

This is a good way to directly access the menus on terminals which do not support the <Alt>-<letter> combinations used on the console. For example, the <ESC> **B** sequence could replace the <Alt>-<N> combination as the “hot-key” to access the {**BLOCK**} menu.

When functions keys are not available, they can be simulated with <Esc> *digit* sequences, e.g. <F1> becomes <Esc> <1>.

When choosing Escape Sequences, it important to avoid using one sequence which is a prefix of another. For example, you cannot use <Esc> <1> to replace <F1> and <Esc> <1> <0> to replace <F10> since the former sequence is a prefix of the latter. When you tried use the latter sequence, VEDIT would recognize and act on the prefix sequence before you’d even typed the <0>.

The usual solution to this particular problem is to use <Esc> <0> to replace <F10>. <F11> and <F12> are harder; one solution is to use <Esc> plus shifted digits so that <F11> becomes <Esc> <!> and <F12> is <Esc> <@>.

Although VEDIT will usually display an error if a new definition is a prefix of another one, it cannot detect all possible problems.

Because the <Esc> key is a prefix of every Escape Sequence, it can by itself not be assigned to the **ESCAPE** function, bringing up the {**ESCAPE**} menu.

The most common solution is to assign <Esc><Esc> to **ESCAPE**, i.e. you must press <Esc> twice to perform the **ESCAPE** function.

Display Layout

Keystroke Equivalent:

(none)

Description:

You can display either the complete keyboard layout or just the Keystroke Macros. The keyboard layout consists of two components:

the basic keyboard layout consisting of the keys assigned to the Edit Functions.

all other assignments are considered Keystrokes Macros.

To display the complete keyboard layout or just the Keystroke Macros:

1. Select **{MISC, Keyboard Layout, Display Layout}**.
2. You are prompted whether you want to display the entire keyboard layout or just the Keystroke Macros. Make your selection.

The layout is displayed one screen at a time; press any key to see the next screen.

Notes: The order in which the keyboard layout is displayed is the same as it occurs in VEDIT's internal "keyboard layout table"; in particular, new Keystroke Macros are always added at the end. You can change the order to one which makes the most sense to you.

To change the display order of the keyboard layout:

1. Select **{MISC, Keyboard Layout, Save to Disk}**.

The "keyboard layout table" will be written to the specified file.

2. Edit the file. Each entry in the keyboard layout is on one line. Put the lines in any order you wish. Save the edited file.
3. Select **{MISC, Keyboard Layout, Load from Disk}**. Specify the filename you just edited. A new "keyboard layout table" will be loaded into VEDIT.
4. Display the layout as before. You may also want to print the layout using **{MISC, Keyboard Layout, Print Layout}**.
5. To make the new "keyboard layout table" permanent in VEDIT, select **{CONFIG, Save to disk}**.

HELP and **{HELP, General}** display the basic keyboard layout in alphabetic order by Edit Function name. This order is not changeable.

See Also: **{MISC, Keyboard Layout}**,
{MISC, Keyboard Layout, Add Keystroke Macro}

Print Layout

Print the entire keyboard layout or just Keystroke Macros.

Keystroke Equivalent: (none)

Description: Instead of just displaying the keyboard layout, you can also print it for a more permanent record.

To print the keyboard layout:

1. Select **{MISC, Keyboard Layout, Print Layout}**.
2. You are prompted whether you want to print the entire keyboard layout or just the Keystroke Macros. Make your selection. The printer settings in **{PRINT, Config}** control the printing.

See Also: All commentary and notes for **{MISC, Keyboard Layout, Display Layout}** also apply here.

Load from File

Load a **.key** file (keyboard layout and macros).

Keystroke Equivalent: (none)

Description: The keyboard layout is determined by VEDIT's internal "keyboard layout table". You can load a new "keyboard layout table" which was supplied by us or which you previously created/saved on disk.

This lets you change the entire keyboard layout "on the fly" and load different sets of Keystroke Macros for different editing tasks.

To load a new "keyboard layout table" file:

1. Select **{MISC, Keyboard Layout, Load from Disk}**.
2. The default filename is **vedit.key**. You can select this name by pressing <Enter> or enter the desired filename, or enter ***.key** for Point & Shoot file selection. The new keyboard layout table will be loaded.
3. To make the new keyboard layout table permanent in VEDIT, select **{CONFIG, Save to disk}**.

Notes: Loading a **.key** file overwrites any previous keyboard layout and Keystroke Macros. If you have created any Keystroke Macros that you want to keep, you must first save them, perhaps by using **{MISC, Keyboard Layout, Save to Disk}**.

Before VEDIT loads the **.key** file, it scans it for validity. If an error is detected, it displays an error message including the line number on which the error occurred. Errors can occur when you edit the file and misspell the name of a key or Edit Function, e.g. you typed “Isnert” or “[CURSR UP]” in place of “Insert” or “[CURSOR UP]”.

See Also: {MISC, Keyboard Layout, Display Layout},
{MISC, Keyboard Layout, Save to Disk}

Save to File

Save a **.key** file (keyboard layout and macros).

Keystroke Equivalent: (none)

Description: The keyboard layout is determined by VEDIT’s internal “keyboard layout table”. You can save the entire “keyboard layout table”, including any new Keystroke Macros you may have added, to a disk file. Since these files normally have a filename extension of **.key**, we often refer to them as just **.key** files.

After saving a **.key** file you can either simply load it back into VEDIT at a later time or edit the file using VEDIT. By editing a **.key** file you can change existing Keystroke Macros, add new Keystroke Macros, change the basic keyboard layout (assignments to the Edit Functions) and determine the order in which the keyboard layout is displayed.

To save the current keyboard layout to disk:

1. Select {MISC, Keyboard Layout, Save to Disk}.
2. The default filename is **vedit.key**. You can select this name by pressing <Enter>, or enter the desired filename, or enter ***.key** for “Point & Shoot” file selection. If the selected file already exists, you will be prompted for confirmation to overwrite the file.

To edit an existing **.key** (keyboard layout table) file:

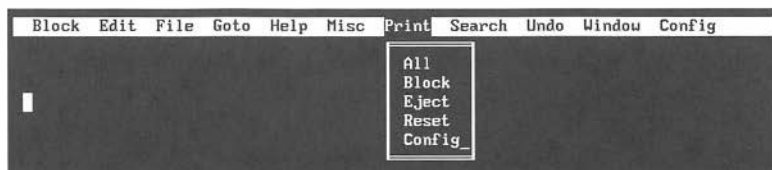
1. Load the file into VEDIT in the same way as you would any other file.
2. Normally, leave the first line alone. In particular, *do not move* the first line.
3. Notice that all remaining lines have the same format. The line begins with the key or keys which you press to perform an editing function. This is followed by one or more spaces/tabs. Then comes the entire editing sequence on one line.
4. As long as each line has the correct format, you can add new lines, delete lines and modify lines. Wherever you need a “tab” character enter “[TAB CHARACTER]”; wherever you need a **newline** enter “[RETURN]”.
5. When done, save the file in the normal way. You can then immediately load it into VEDIT with
{MISC, Keyboard Layout, Load from Disk} for testing.

See Also: **{MISC, Keyboard Layout, Display Macros}**
{MISC, Keyboard Layout, Load from Disk}

Print Menu

Keystroke Equivalent: <Alt>-<P>, This is a Keystroke Macro.

Introduction: You can print the file being edited on a local or network printer. You can either print the entire file or any portions of it. When printing, a “*printer margin*” is included to keep the text from printing on the very left edge of the paper. VEDIT normally also skips over page perforations and leaves a top and bottom margin on each page. These and other parameters are adjustable.



All Print the entire file.

Keystroke Equivalent: (none)

Description: This item prints the entire file currently being edited. For large files, this may take some time. Formatting is controlled by the configurable print parameters described below.

The message “PRINTING - Press <Ctrl>-<C> to abort” appears on the status line while VEDIT is printing. At any time you may abort the printing by pressing <Ctrl>-<C>.

This only stops output from VEDIT to QNX. Since some printers have buffers and QNX may be spooling text which is to be printed, printing may continue for some time after VEDIT stops sending text.

Notes: After the entire file has been printed, this item advances the printer to the top of the next page. Use **{PRINT, Eject/Flush}** to feed additional blank pages through the printer.

On a network printer, this item also releases (flushes) the print job.

If you abort the printing by pressing <Ctrl>-<C>, you may notice that the cursor has moved. You can restore the cursor position by selecting **{UNDO, Edit}** (<Alt>-<F7>).

See Also: **{PRINT, Block}**, **{PRINT, Eject/Flush}**

Block Print a highlighted block of text.

Keystroke Equivalent: <Ctrl>-<P>, This is a Keystroke Macro.

Description: Instead of printing the entire file, you can print just a highlighted block of text. Since this item only prints the block and does not advance the printer to the top of the next page, you can print several blocks of text on the same page.

To print a block of text:

1. Assuming no block markers are set, set the first block marker by selecting either **{PRINT, Block}** (<Ctrl>-<P>) or **{BLOCK, Set Marker}** (<F9>). The cursor should normally be in the first column of the first line to print.

2. Move the cursor to highlight the desired block in the normal way. Remember to position the cursor one character *past* the last character to be printed.

Since you normally print entire lines, position the cursor in the first column of the line following the last line to be printed.

3. Select **{PRINT, Block}** (<Ctrl>-<P>). The block should begin printing.

Notes: When no block markers are set, this item performs the same function as **{BLOCK, Set Marker}**.

On “page” printers such as laser printers, the entire block will not physically print until you select **{PRINT, Eject/Flush}**.

On network printers, nothing will physically print until you select **{PRINT, Eject/Flush}** or exit VEDIT.

If your block does not include the **newline** character at the end of the last line, the last line will not print and any following printed text may get appended. See “Marking (selecting) a Block of Text in Chapter 3 (Editing Guide) if you have trouble getting the correct amount to print.

Don’t manually advance the carriage on line printers in order to remove a printed page; instead advance the paper with **{PRINT, Eject/Flush}**.

Since VEDIT keeps track of where on the page it is printing, advancing the carriage manually will cause VEDIT to print on page perforations. If the VEDIT and the printer do get out of sync, manually advance the printer to the top of a page and then select **{PRINT, Reset}**.

See Also: “Block Operations” in Chapter 3 (Editing Guide).
{PRINT, All}, **{PRINT, Eject/Flush}**, **{PRINT, Reset}**

Eject/Flush

Advance the printer to the top of the next page; “flush” (release) the print job on network printers.

Keystroke Equivalent: (none)

Description: This item advances the printer to the top of the next page. It is often used to feed out a page that was printed with **{PRINT, Block}**.

Depending on the setting of **{PRINT, Config, Enable Form-Feed}**, VEDIT will either advance the paper by using the correct number of “Line-Feed” characters or with a single “Form-Feed” character.

On network printers, this item also releases (flushes) the print job so that it will begin to actually print.

See Also: Notes for **{PRINT, Block}**.
{PRINT, Config, Paper length},
{PRINT, Config, Enable Form-Feed}

Reset

Reset VEDIT to assume the printer is at the top of a page.

Keystroke Equivalent: (none)

Description: VEDIT keeps track of the number of lines which have been sent to the printer so that it knows when the printer has reached the bottom margin and should be advanced to the next page.

{PRINT, Eject} lets you reset VEDIT’s line counter when the printer is at the top of a page. You will need this if you manually advance the paper in your printer. It is preferable to feed paper through your printer with **{PRINT, Eject/Flush}**.

To re-synchronize VEDIT with your printer:

1. Advance your printer to the top of a new page. This is best done by pressing “Form-Feed” on your printer, and if the printer is not at the top of a new page, turning the carriage until it is.
2. Select **{PRINT, Reset}**.

VEDIT should now be able to print properly again.

Notes: If VEDIT continues to print on page perforations and/or leaves blank space in the middle of a page, you should check the settings for **{PRINT, Config, Paper Length}** and **{PRINT, Config, Enable Form-Feed}**.

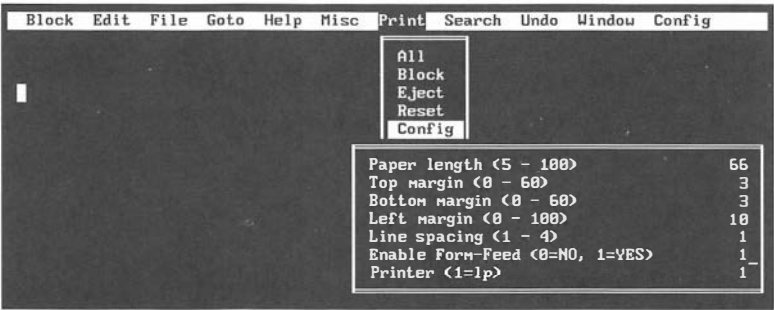
See Also: Notes for **{PRINT, Block}**.
{PRINT, Config, Paper length}

Config
(Submenu)

Set paper size, margins, line spacing and enable Form-Feeds.

Keystroke Equivalent: (none)

Description: VEDIT can be configured for use with a wide variety of printers and paper sizes. Selecting **{PRINT, Config}** displays the following submenu:



Notes: If you set all three printer margins to “0”, VEDIT will print the text “raw” without any additional formatting. This is useful for printing a file which has already been formatted by another program (e.g. Compiler listing file, etc).

Paper length (5 - 100)
[Default = 66]

This value should be set to the length of the paper in lines. Since most printers print 6 lines per inch and standard paper is 11 inches long, this value is usually set to 66. You will need to change this value when printing on legal size paper (80 lines), European sized paper or pre-printed forms. You may want to change it when printing labels.

The number of lines of text printed per page is equal to the “Page length” minus “Top margin” minus “Bottom margin”. The default is 60.

Notes: Many laser printers have a default setting of 62 lines per page. This will work properly with a “Paper length” of 66 as long as **{PRINT, Config, Form-Fed}** is set to “Yes” and the top/bottom margins are set such that you aren’t attempting to print more than 62 lines of text per page.

Top margin (0 - 60) [Default = 3]	This value determines how many lines are left blank at the top of each page.
Bottom margin (0 - 60) [Default = 3]	This value determines how many lines are left blank at the bottom of each page.
Left margin (0 - 100) [Default = 10]	This value is set to avoid printing at the very left edge of the paper. The default value of “10” leaves about a one inch margin. Setting this value has no effect on editing, it has the same effect as adjusting the paper sideways in the printer.
Notes:	This “Left margin” is added to any left margin in your text. For example, if your text was written using a left margin of 8 characters and you print this document with a left margin for the printer of 10 characters, total space to the left of each line will be 18 characters.
Line spacing (1 - 4) [Default = 1]	This value determines whether the text is printed single, double or triple spaced. A value of 1 is normal and a value of 2 would leave a blank line between every printed line.
Enable Form-Feed [Default = Yes]	Most printers can be advanced to the top of the next page by sending them a single “Form-Feed” character. Otherwise, VEDIT issues the correct number of “Line-Feeds” to advance to the top of the next page. When possible, it is better to set this value to “Yes”; the “Paper length” value is then not so critical.
Printer (0=/dev/par) [Default = 0]	This value determines to which parallel or serial port VEDIT prints. The default is lp .

Search Menu

Keystroke Equivalent: <Alt>-<S>, This is a Keystroke Macro.

Introduction: You can easily and very quickly search for a particular string of text and optionally replace it with different text.



Search

Search for a string of text.

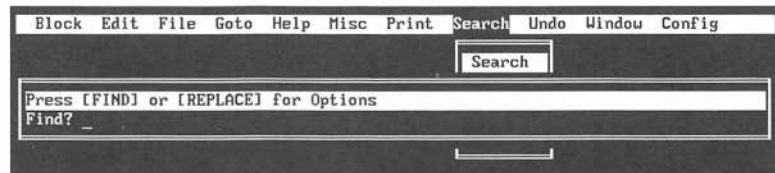
Keystroke Equivalent: <F2>, This is identical to the Edit Function **FIND**.

Description: VEDIT can search your file for the “*search string*” that you specify. When needed, you can select from the “*search options*”. Options include searching backward or searching only for entire words.

Besides searching for a “literal” sequence of characters, you can also search using VEDIT’s powerful *pattern matching*. This is fully described in Chapter 3 (Editing Guide).

To perform a simple search:

1. Select **{SEARCH, Search}** or press **FIND (<F2>)**. You are prompted for the “search string”:



2. Enter the exact sequence of characters you would like to locate.

The prompt “Find?” indicates you are in “short form pattern matching” mode. In this mode the only special character is “|”; you must enter “||” to search for a single “|”.

A prompt of “Find (Expression)?” indicates you are in “regular expression pattern matching” mode; in this mode many characters have a special meaning.

If found, the cursor will be positioned just past the first occurrence of the located text. If the search string was not found, the status line will display the error “CANNOT FIND: *search string*”.

3. To search for the next occurrence of the same string either select **{SEARCH, Again}** or press **FIND AGAIN (<F3>)**.

HINTS:



Remember that VEDIT lets you recall previous search strings by pressing **CURSOR UP**.

You can also edit the search string which you are entering or which you recalled.



The online help for searching is quite extensive. It explains how to search for special characters such as **newline**, **null**, control and graphics characters.

Search Options

VEDIT normally searches from the current cursor position forward toward the end of the file. Other types of searching are possible by selecting one or more of the following “search options”:

[A]gain

Re-use the previously entered search string. It is often used in conjunction with “Begin” to restart the search from the beginning of the file, or with “Reverse” to restart the search in the reverse direction.

[B]egin

Start searching from the beginning of the file; otherwise the search starts from the current cursor position. (You may want to first set a Text Marker at the current cursor position if you want to return there after the search.)

[F]ar

This option only appears in the unusual case that you have set **{CONFIG, File Handling, Enable global file operations}** to “No”. This option causes the search to continue to the end of the file, if necessary, instead of just to the end of the file portion currently in memory. This option is the **DEFAULT** when the above configuration switch is set to the usual “Yes”.

- [N]ear** This option restricts the search to the portion of the file currently in memory. It also causes the **Begin** option to start searching from the beginning of the text currently in memory. It has rather limited use.
- [R]everse** Search from the cursor position toward the beginning of file. This options effectively searches for the previous occurrence, rather than the next.
- [W]ord** Restricts the search to entire “words”. This way a search for “the” will not find “there”, “other” and similar words.
- Expression [Y/N]** Selects whether or not “Regular Expressions” should be used for the search (and/or replace). It overrides the current search mode set by **{SEARCH, Options, Regular expression search mode}**. Note that when Regular Expressions are enabled, the search prompt changes to “Find (Expression)?”.

To search using one or more “search options”:

1. Select **{SEARCH, Search}** or press **FIND (<F2>)**. You are prompted with “Find?” (or “Find (Expression)”).
2. Press **<F2>** or **<Alt>-<F2>** to bring up the search options:

Press **<F2>** or **<Alt-F2>** for Options

Find?

OPTIONS: **[A]gain [B]egin [N]ear [R]everse [W]ord**
Expression[Y/N]

3. Select one or more of the options by typing the corresponding letter (enclosed by “[]”). Press **<Enter>** to accept the options. For example, enter “RW” to search backwards for entire words.
4. Unless you selected the “Again” option you will be returned to the “Find?” prompt. Enter the search string as usual.

Searching within a block

You can restrict the search (or replace) to a highlighted block. This feature is primarily useful for search & replace operations that you only want to make in a portion of the file.

To search within a block of text:

1. Highlight the desired text; you can highlight either a normal character block or a columnar block. You must set both block markers—“BLOCK” should be displayed on the status line.
2. Assuming you want to search forwards in the block, position the cursor at the beginning of the highlighted block. The search starts at the cursor position, not necessarily at the beginning of the block.

Alternatively, you can position the cursor at the end of the block and use the “Reverse” search option to search backwards within the block.

3. Select **{SEARCH, Search}** or press **FIND (<F2>)**. You can now enter the search string and/or select search options in the normal way.

You must use extra care when searching (or replacing) within a block of text. Other editors you may have used simply force the search to the beginning of the block after you highlight the text. However, we wanted to give you additional flexibility so that you can highlight a block and also search outside the block, e.g. search for the location to which you want to copy the block.

The following rules govern searching within blocks:

1. If the cursor is within the highlighted block when you select **{SEARCH, Search}**, **{SEARCH, Replace}**, **FIND** or **REPLACE**, the search is restricted to the block; otherwise the search operates normally within the file.
2. The search always starts at the current cursor position and not automatically at the beginning of the block.

3. The entire search string must be found within the highlighted text; this consideration is especially important when searching within columnar blocks. (In other words, VEDIT will not locate a word where half of it is highlighted and the other half is not highlighted.)

Notes: By default, searches are not “Case Sensitive”, which means that searching for “This” will also locate “THIS” and “this”. You can specify case sensitive searching by setting **{SEARCH, Options, Equate Upper/Lower case during search}** to “No”.

In short form pattern matching mode, to search for the special character “|”, enter ||. In regular expression mode many characters have a special meaning; to search for one of them, precede it with “\”.

The search and replacement strings are limited to a single line of 80 characters.

To search for text that includes a “newline” press <Ctrl>-<N>; this will display “<CR><LF>” to indicate the **newline** characters. (The <Ctrl>-<N> is not configurable; think “N” for **newline**.)

To search for a control character, precede it with **ENTER CTRL** (<Ctrl>-<Q>). This forces the next character to be literally added to the search string.

To search for graphics characters with values between 128 and 255, on a console, press and release <Alt> then type the hexadecimal value of the character.

VEDIT has a special way of searching for the null character, hex 00, the byte with all bits zero. In VEDIT’s *short form* pattern matching mode, |0 indicates the null character; in *regular expression* mode, \000 indicates the null character.

VEDIT positions the cursor past the located text; some other word processors position the cursor at the beginning of the located text.

If you don’t use the default keyboard layout, the top line of the dialog box will display the current assignments for **FIND** and **REPLACE** needed to access the search options.

{SEARCH, Search} is primarily only selected with a mouse, because it is easier to simply press **FIND** (<F2>).

See Also: “Search and Replace” in Chapter 3 (Editing Guide).

{SEARCH, Again}

FIND, FIND AGAIN

Replace

Locate a text string and replace it with another text string.

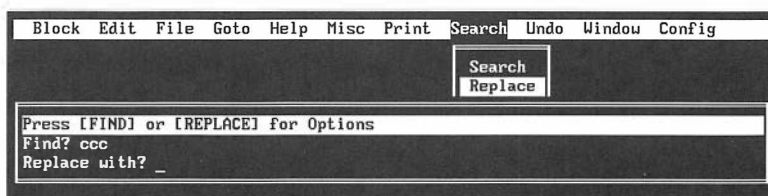
Keystroke Equivalent: <Alt>-<F2>, This is identical to the Edit Function **REPLACE**

Description: With **{SEARCH, Replace}** or **FIND** (<F2>) you can search your file for a text string and replace it with another. You can perform the replacement a single time, globally throughout the file or selectively.

To perform a simple search and replace operation:

1. Select **{SEARCH, Replace}** or press **REPLACE** (<F2>).

You are first prompted for the “search string”: (You will initially only see the first prompt.)



2. Enter the “search string” in the same way as for **{SEARCH, Search}**. Press <F2> or <Alt>-<F2> to access the search options if needed.
3. Enter the “replacement string” exactly as you want it in the text. Lower case letters are not converted to upper case, nor do they match the case of the original text. (If you need this capability you will have to use “Regular Expressions”). If necessary, you can still access the search options at this time.

If found, the cursor will be positioned just past the first occurrence of the located text and the status line will prompt you with:

REPLACE? [Y]es [N]o [R]est [O]ne [C]ancel

Select the desired option by pressing the corresponding letter:

[Y]es	Replace the text for this occurrence and immediately search for the next occurrence.
[N]o	Don't replace the text for this occurrence; immediately search for the next occurrence.
[R]est	Replace the text for this and all subsequent occurrences. VEDIT will perform the remaining replacements without prompting.
[O]ne	Replace the text for this occurrence and return to normal editing.
[C]ancel	Don't replace the text for this occurrence. Cancel the replace operation; return to normal editing.

Notes: The description and notes for **{SEARCH, Search}** apply here too.

With VEDIT's short form pattern matching, located text cannot be part of the replacement text. However, the "regular expression" search mode has this capability. For example the regular expression search string **{ { [Hh] } e l l o }** will locate "Hello" or "hello". The replacement string **\1i** will replace "Hello" with "Hi" and "hello" with "hi". Regular expressions are fully described in Chapter 3 (Editing Guide).

{SEARCH, Replace} is used mainly with a mouse. At the keyboard simply pressing **REPLACE** (<Alt>-<F2>) is easier than accessing the menu.

See Also: **{SEARCH, Search}**, **{SEARCH, Again}**
REPLACE

Again Repeat the last "Search" or "Replace" operation.

Keystroke Equivalent: <F3>, This is identical to the Edit Function **FIND AGAIN**

Description: **{SEARCH, Again}** and **FIND AGAIN** perform the last “Search” or “Replace” operation one more time. It will use the same search options that may have been specified for the original search/replace. It will only be restricted to the highlighted block of text if the original search/replace was also so restricted.

Notes: You can recall previous search and replace strings by pressing **CURSOR UP** when you are prompted for these strings.

See Also: **{SEARCH, Search}**, **{SEARCH, Replace}**

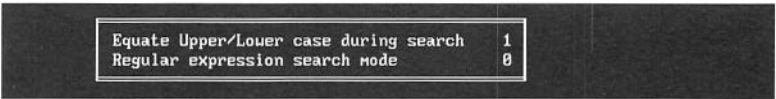
**Config
(Submenu)**

Change “Search” and “Replace” options.

Keystroke Equivalent: (none)

Description: Besides the “search options” described under **{SEARCH, Search}**, several additional search and replacement configuration options are available. After changing these configuration options, you can make them permanent by selecting **{CONFIG, Save to disk}**.

When you select this item, the following submenu will appear:



Equate Lower/Upper
case during search
[Default = Yes]

When this parameter is set to “Yes”, VEDIT will ignore the case of a letter when it is searching. For example, the search string “this” will also locate “This” and “THIS”. When set to “No”, the letters of the search string must match the case of the located text.

Regular expression
search mode [Default =
Yes]

VEDIT is always in one of two possible search modes. When this parameter is set to “Yes”, VEDIT is in the “Regular Expression” search mode. When set to “No”, VEDIT is in its short form pattern matching search mode.

You can temporarily change the search mode using the “search options”.

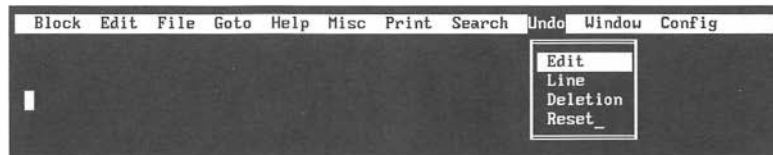
These search modes are fully described in Chapter 3 (Editing Guide).

See Also: **{SEARCH, Search}**, **{SEARCH, Replace}**

Undo Menu

Keystroke Equivalent: <Alt>-<U>, This is a Keystroke Macro.

Introduction: VEDIT remembers each edit operation performed and can quickly undo them in reverse order including cursor movement; insertions, deletions and search/replace. You can back up through these operations keystroke by keystroke, line by line or deletion by deletion. Not only does this let you back out of mistakes, it gives you the luxury to try out changes to your file and “undo” them if they don’t work out.



As supplied, VEDIT remembers the last 500 edit operations in *each* file being edited. You can configure VEDIT to remember up to the last 1000 operations when enough memory is available by editing the `.cfg` file. for details, see Chapter 7 (Configuration).

As you continue to work and exceed the maximum number of edit operations VEDIT can remember, the oldest operations are forgotten. Because of this, you can only go as far back as 1000 keystrokes.

Notes: Each basic edit operation consumes one “undo level”; VEDIT is typically configured for 500 undo levels. Therefore, VEDIT could undo the last 500 cursor movements, characters typed in, or single characters deleted. However, some editing functions consume many undo levels; therefore, you can undo far fewer of these. These functions (and the number of undo levels used) include: columnar block operations (3 levels per line), paragraph formatting (3 levels per line), global search and replace (3 levels per replacement). Obviously, Keystroke Macros can also consume many undo levels.

In extreme cases, a single editing operation, e.g. a Keystroke Macro, may be so complex that it consumes more than the available undo levels. Such operations can not be undone.

When deleting large blocks of text, VEDIT must save this text in memory in case you want to “undo” the deletion. VEDIT allocates at most 64K for deleted text. If you delete either a single large block of text or successive smaller blocks of text which exceed VEDIT’s undo storage, you will get the following confirmation prompt:

OK to delete xxxx lines and reset undo (Y/N)

This is a warning that VEDIT won’t be able to undo the deletion if you go ahead and confirm the deletion. This also resets the undo system.

See Also: “Undo Memory Management” in Appendix B (Memory Management).

Edit

Back up a single keystroke or editing operation.

Keystroke Equivalent: <Alt>-<F7>, This is a Keystroke Macro.

Description: Each time {**UNDO, Edit**} is used, it reverses the effect of the previous edit operation. It can be used repeatedly to back up step by step.

Notes: Selecting a menu item or performing a search/replace is considered a single edit operation even though it may take several keystrokes to perform.

If you undo all the way back to the point where you loaded the file, VEDIT will know that the file has not been altered.

Line

Undo all changes made to the current line.

Keystroke Equivalent: <Ctrl>-<U>, This is a Keystroke Macro.

Description: All changes made to the current line of text can be reversed in one step with {**UNDO, Line**}.

Immediately selecting {**UNDO, Line**} again will move the cursor to the previously modified line, but not yet undo any changes to this line. This lets you review whether you also want to undo this line. Select {**UNDO, Line**} again to also undo this line.



{**UNDO, Line**} should be used with caution; it can rapidly undo an appreciable amount of work. VEDIT has no “Redo” feature to reverse “Undo”. Therefore, if by mistake your “Undo” actions remove text that you wanted to keep, there is no way to recover that text short of retyping it.

Deletion

Insert the last text deletion(s).

Keystroke Equivalent:

(none)

Description:

Up to five blocks of text deleted with **ERASE EOL**, **ERASE LINE** and {**BLOCK, Delete**} can be “undeleted” or re-inserted at any desired place in your file using {**UNDO, Deletion**}.

This works independently of other undo operations. Even if other undo functions have been used to restore these blocks of text exactly where they originally resided, {**UNDO, Deletion**} can re-insert the same blocks at the current cursor position.

Blocks are re-inserted in the reverse order they were deleted (technically “popping” the text off a deletion “stack”).

{**UNDO, Deletion**} can be used as a quick way to move a line of text.

A quick way to move a line to text:

1. Position the cursor on the line to be moved and delete it with **ERASE LINE** (<Ctrl>-<X>).
2. Move the cursor to the beginning of the line in front of which you want the deleted line re-inserted.
3. Select {**UNDO, Deletion**}.

The line has effectively been moved to its new position.

Notes:

Technically, {**UNDO, Deletion**} is not an undo function; it is an editing function which inserts a previously deleted block of text anywhere in the file. Therefore, you can undo {**UNDO, Deletion**} with {**UNDO, Edit**}.

When enough memory is available, VEDIT saves up to 64K of deleted text for undo purposes. Only the last five deletions of three or more characters can be re-inserted with **{UNDO, Deletion}**.

If the last five deletions exceed 64K, it may not be possible to re-insert more than one or two deleted blocks.

A block deletion giving the confirmation prompt “OK to delete xxxx lines and reset undo (Y/N)” cannot be re-inserted or otherwise undone.

Reset

Reset the undo memory.

Keystroke Equivalent:

(none)

Description:

{UNDO, Reset} clears the undo memory (and resets all undo stacks) so that you can later not undo any further this point in your editing. You can then try out different editing scenarios and if they do not work, you can use **[REPEAT] 255 {UNDO, Line}** to quickly undo back to this point.

Window Menu

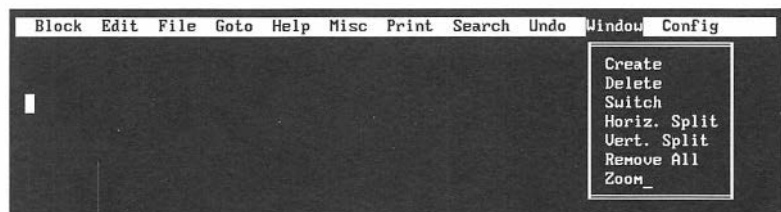
Keystroke Equivalent:

<Alt>-<U>, This is a Keystroke Macro.

Introduction:

The **{WINDOW}** menu gives you additional flexibility to create windows of the exact size you want. For example, you might want to have a large window at the top of the screen for the main file you are editing and a small window at the bottom for a reference file or additional notes.

{WINDOW} also lets you delete windows, zoom the current window to fill the full screen and display a file in multiple windows.



See Also: “Windows” in Chapter 3 (Editing Guide).



Throughout this manual, the term “window” refers to a **VEDIT** window. VEDIT creates and manages its own vertical and horizontal windows within whatever text display facility QNX makes available to it. These are the VEDIT windows we refer to.

The QNX text display facilities used may be of several types. Perhaps the simplest case conceptually is running VEDIT on a terminal. In this case the terminal screen is the VEDIT text display and that’s the whole story (except of course for terminal set up—see **terminfo** in the *Utility Guide*).

On the system console, the situation can become a bit more complex giving the user more options. QNX supports multiple *virtual consoles*, making the single physical console appear to the system as many consoles. Each virtual console runs its own application(s) independently of other consoles. The user switches among virtual consoles with special key combinations such as **<Ctrl>-<Alt>-<gray +>**.

One instance of VEDIT needs one virtual console. It is possible to have several instances of VEDIT running independently on different virtual consoles. Programmers often use multiple virtual consoles, editing on one (or more) and compiling on another. Some extend this—testing the compiled program on another, running the debugger on yet another, and so on.

Create

Creates a new window.

Keystroke Equivalent:

(none)

Description:

{WINDOW, Create} creates a new window by splitting the current window into two. A horizontal window at the top or bottom, or a vertical window at the right or left side of the current window can be created.

To create a new window on the screen:

1. If necessary, use **{WINDOW, Switch}** to switch to the window which is to be split into two windows. Select **{WINDOW, Create}**.

The following dialog box prompts you for the necessary information: (You will initially only see the first prompt.)



2. Respond to the first prompt by pressing “T”, “B”, “L” or “R” to select where in the current window you want the new window created.
3. For “Enter Window Name” type the desired single character name for the new window. This is generally the name of the edit buffer to be displayed in the window, e.g. “A”.
4. The third prompt is “Enter # Columns” or “Enter # Lines” depending upon whether you are creating a vertical or horizontal window. Enter the desired number of columns/lines for the new window. The remaining columns/lines will be used for the existing window.

Notes: @ is the name of the initial window on startup; it is attached to the main edit buffer @.

A newly created window will usually remain empty until you switch to an edit buffer with the same name as the new window.

Windows may be as small as one line and/or 10 columns. Minimal borders are drawn on the top and left to separate windows from each other. Overlapping windows are not supported.

VEDIT supports separate colors for each window.

{MISC, Color Toggle} toggles the current window through different colors.

The VEDIT macro language has the ability to create borderless windows; it also has many other windowing capabilities.

See Also: **{FILE, Open File (new buffer)}**
{WINDOW, Switch}, **{WINDOW, Horiz. Split}**,
{WINDOW, Vert. Split}

Delete

Delete a window.

Keystroke Equivalent: (none)

Description: **{WINDOW, Delete}** simply prompts for the name of the window to be deleted and removes it from the screen. Other window(s) expand in size to reclaim the screen space.

This function *does not affect any files!* The edit buffer associated with the deleted file can still be edited and viewed in another window, normally the main window @.

Notes: You cannot delete the main window @.

When the window is deleted, VEDIT switches to the main @ edit buffer (file) and window.

If you want to remove all windows at one time, select **{WINDOW, Remove All}**.

Here is how the existing windows will reclaim the screen space:

VEDIT remembers the commands (location, name and size) for all created windows. When you delete a window, VEDIT resets the screen and then replays the window creation commands, leaving out the window which was just deleted.

See Also: **{WINDOW, Remove All}**

Switch

Switches to another window and/or edit buffer.

Keystroke Equivalent:

<Alt>-<F4>

Description:

{WINDOW, Switch} prompts with:

Enter Window / Edit Buffer name:

If you type the name of an existing window, VEDIT switches to that window *and* to its associated edit buffer. If you type the name of a non-existent window, VEDIT switches to that edit buffer; in this case it is identical to **{FILE, Buffer Switch}**.

{WINDOW, Switch} is primarily used for switching between the multiple windows used to display one file, i.e. after you have used **{WINDOW, Vert. Split}** or **{WINDOW, Horiz. Split}**.

If you don't have multiple windows per file, it is preferable to use **{FILE, Buffer Switch}** or the built-in Keystroke Macro <Alt>-<T>.

Notes:

You can switch to the main edit buffer and default window by simply pressing <Enter> at the prompt.

You can switch to windows which aren't displayed on the screen while the current window is zoomed to full screen. The switch will un-zoom the screen.

Mouse users may find it easier to switch windows simply by clicking the mouse anywhere in the desired window. (Currently you can't do this if a window is zoomed to full screen.)

When switching to another window on the same file, there may be some delay while VEDIT brings the new portion of the file to be edited into memory. The new window becomes the "primary attached" window for the corresponding edit buffer.

See Also:

{FILE, Buffer Switch}
Other **{WINDOW}** items.

**Horiz. Split
Vert. Split**

Splits the current window into two horizontal or vertical windows which display the same file.

Keystroke Equivalent:

<Alt>-<V>, This Keystroke Macro performs a vertical split.

Description: Sometimes you want to display a file in two or more windows. This lets you see and edit two portions of the file without having to manually scroll back and forth. For example, you may want to refer to the definitions listed at the beginning of a file while you are editing in the middle of the file.

{WINDOW, Horiz. Split} creates a new window in the lower half of the current window for displaying the same file.

{WINDOW, Vert. Split} creates a new window in the right half of the current window for displaying the same file.

VEDIT automatically assigns each such window a unique name—"0", then "1" and so on. The current file will immediately be displayed in the new window.

Once created, use **{WINDOW, Switch}** to switch to the new window and examine or edit another part of the same file. Each window on the same file can be scrolled independently of the other window(s). A separate cursor position is maintained in each window.

Notes: VEDIT always keeps all windows on the same file up to date. When the windows overlap, you will notice your edit changes occurring in both windows.

By restricting your edit buffers to "A" through "Y" and leaving "0" through "9" as simple Text Registers, you will avoid potential edit buffer/ window name conflicts with the windows "0", "1", etc. created by these functions.

The built in Keystroke Macros <Alt>-<Y> and <Alt>-<V> create horizontal windows for viewing multiple files, and vertical windows for viewing the same file in multiple windows. You can use the **{FILE, Open File (new buffer)}** function and the **{WINDOW}** menu to create other combinations.

It is not possible to create custom sized windows for viewing the same file. (Macros written in the VEDIT macro language could do this.)

See Also: "Windows" in Chapter 3 (Editing Guide).

Remove All

Removes all windows and initialize the screen.

Keystroke Equivalent: (none)

Description: This item removes all windows from the screen; only the default @ window will remain and will cover the full screen.

The status of all edit buffers remains unchanged! No text is lost!

Zoom

Zoom or un-zoom the current window.

Keystroke Equivalent: (none)

Description: When you have created several windows on your screen, it is sometimes helpful to dedicate the whole screen to a particular window for easier editing.

{WINDOW, Zoom} expands the current window so that it fills the entire screen, thereby displaying more of the current file. The zooming remains in effect until you switch to another window or select {WINDOW, Zoom} again to “un-zoom” the screen to redisplay all windows.

See Also: “Windows” in Chapter 3 (Editing Guide).
{WINDOW, Switch}

Config Menu

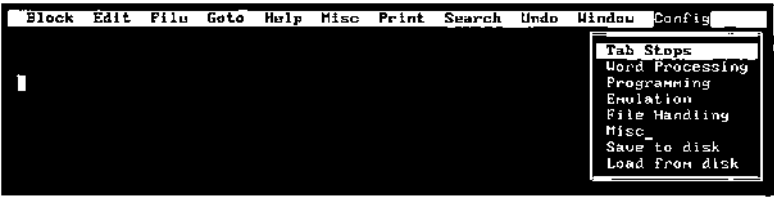
Keystroke Equivalent: <Alt>-<C>, This is a Keystroke Macro.

Introduction: The Config menus lets you tailor VEDIT to your applications and personal preferences. Any configuration changes you make remain temporary unless you explicitly make them permanent with {CONFIG, Save to disk}.

NOTE:



Not all configuration options are in this menu! Printer related configuration is in {PRINT, Config}, search related configuration is in {SEARCH, Config} and keyboard layout related configuration is in {MISC, Keyboard Layout}.



Most Config Menu items are submenus of configuration settings. When you browse through the Config Menu using <↑> and <↓>, you will see the contents of each submenu displayed. This lets you see settings available in each submenu before you actually select it.

Notes: A great deal of experience has gone into developing the default settings for VEDIT. We strongly suggest that you at least try our settings before making numerous changes.

Tab Stops

Displays and changes the tab positions.

9	17	25	33	41	49	57	65
73	81	89	97	105	113	121	129
137	145	153	161	169	177	185	193
201	209	217	225	233	241	249	254

Change Tab positions (Y/N)?

If you type “Y” to change the tab positions, you are prompted with:

Enter the Tab interval -OR- enter up to 33 tab positions

If you only enter one number, it is considered the tab interval. If you enter more than one number, they are considered to be explicit tab positions.

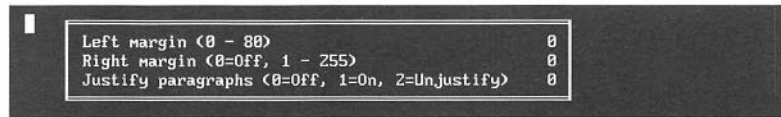
Since the far left column is column number 1, the first tab stop will be at the interval plus 1. The display above was set using an interval of 8.

If for some reason you only wanted a single tab stop, for example “20”, you could enter it as “20 20”.

Any Tab characters in your file past the last tab stop are displayed as “^I”.

Word Processing (Submenu)

Permits changing the left and right margins and enabling justification.



Left margin (0 - 80)
[Default = 0]

Sets the left margin for use with paragraph formatting and indentation.

When VEDIT starts up, it always resets this value to 00; in other words you can only temporarily and not permanently change this value.

Do not confuse this parameter with **{PRINT, Config, Left margin}** which is the left margin used only for printing.

Right margin (0=Off, 1 - 255) [Default = 0]

Sets the right margin for word wrap and paragraph formatting. Typical values are between 50 and 80. A value of "0" disables both word wrap and formatting.

It should be disabled when editing programs!

Justify paragraphs
(0=Off, 1=On,
2=Unjustify) [Default = 0]

Determines whether **{EDIT, Format Paragraph}** will also justify or unjustify paragraphs.

No Disables justification

Yes Enables justification.

Unjustify "Unjustify" the paragraph, removing extra spaces.

Programming (Submenu)

Provides settings for program-writing aids.

```
Auto-Indent Mode          0
Indent Increment (1 - 20) 4
Lower/upper case key conversion (0 - 3) 0
Conditional conversion character (32 - 126) 59
```

Auto-Indent Mode
[Default = No]

When “Auto-Indent Mode” is on, the indent position for a new line of text is initially the same as for the previous line of text. This is convenient for C programming. The indent position can then be changed with **INDENT** and **UNDENT**.

Indent increment (1 - 20)
[Default = 4]

Determines how much VEDIT indents/undents each time **INDENT** or **UNDENT** is pressed.

Lower/Upper case key
conversion (0 - 3)
[Default = 0]

Determines whether lower case characters typed on the keyboard are converted to upper case. It is primarily for assembly language programming. It does not affect any existing text; use **{EDIT, Lower/Upper case}** to convert existing text.

The possible values are:

- 0 No conversion takes place.
- 1 All lower case letters are converted to upper case.
- 2 Lower case letters are converted to upper case, unless the cursor is past the “Conditional conversion character” (below). Primarily for assembly language programmers that desire having the Label, Opcode and Operand in upper case and the comment in upper and lower case.
- 3 Similar to (2) except that characters are reversed instead of being forced to upper case.

Conditional conversion
character (32 - 126)
[Default = ;]

Sets the special character that VEDIT looks for when deciding whether to convert lower case keyboard characters to upper case when “Lower/Upper case key conversion” is set to 2 or 3.

Emulation (Submenu)

Contains settings for emulating other text editors and word processors.

```

Cursor positioning mode (0 - 4)      1
Typewriter emulation mode (0 - 2)   2
Expand <TAB> key with spaces         0
Auto-Cancel [FIND] and [REPLACE]    1
Auto-Cancel Block Markers           1
Horizontal scroll margin (40 - 255)  210
Horizontal scroll increment (1 - 100) 20
Retab during block-column insert    1
Trim spaces after block-column insert 1
Advance cursor past block insert     1

```

Cursor positioning mode
(0 - 4) [Default = 1]

This controls how the cursor moves on the screen where there is no text.

In general the cursor only moves to where there is text, avoiding empty parts of the screen. For example, pressing **CURSOR RIGHT** with the cursor at the end of a line moves the cursor to the beginning of the next line.

- 0 The cursor can never be positioned past the end of a line. For example, if you move the cursor down from the end of a long line to a shorter line, the cursor will also move left to the end of the shorter line.
- 1 The cursor can be moved straight up and down from a long line past short lines to another long line. If any attempt is made to change the text with the cursor past the end of a line, e.g. typing in new characters, the cursor first moves left to its “correct” position.
- 2 The cursor “zig-zags” as it is moved up or down past the end of a short line. This mode ensures that the cursor is always located over real text yet preserves the horizontal position from which it started.
- 3 The cursor moves identically to Mode 1. However, if the cursor is past the end of a line and you type new text, spaces are automatically inserted from the end of the line up to the newly entered text. This mode is handy for filling out tables and other formatted text. Note that using this mode may result in unwanted spaces being inserted into the text, thus consuming additional memory and disk space.

- 4 Similar to Mode 3 except that **CURSOR RIGHT** can move the cursor past the end of a line; it does not move to the beginning of the next line. This is the mode to use for totally free-form text layout.

Modes 1 or 2 are recommended for programmers, modes 3 or 4 for word processing. Other word processors generally operate in one of the modes listed and you may want to pick one that you are already familiar with.

Typewriter emulation
mode (0 - 2) [Default = 2]

Sets the behavior of **TAB CHARACTER** and the <Enter> key.

- 0 The **TAB CHARACTER** function inserts a Tab character at the cursor position and advances the cursor. The <Enter> key splits the current line or adds a new one at the cursor position.
- 1 In Overstrike mode, **TAB CHARACTER** moves the cursor to the next tab stop, the same as **TAB CURSOR**. In Insert Mode or at the end of a line, **TAB CHAR** inserts a Tab character. The <Enter> key operates identically to mode 0.
- 2 **TAB CHARACTER** operates identically to mode 1. In Overstrike mode, <Enter> moves the cursor to the next line, same as **NEXT LINE**. In Insert Mode it splits the current line or adds a new one.

A little experimentation is best for understanding these modes and deciding which you like best.

Expand <tab> key with
spaces [Default = No]

Determines whether the <tab> key (**[TAB CHARACTER]** function) is expanded with spaces to the next tab stop. If not, a Tab character is inserted into the edit buffer.

This does not affect any existing text. Use the supplied "TAB-XPND" macro to expand tab character in a file to spaces.

Auto-Cancel Block
Markers [Default = Yes]

Determines whether the new block of text is highlighted following **{BLOCK, Copy to Cursor}** or **{BLOCK, Move to Cursor}**.

- No The new block of text is highlighted. This emulates WordStar, Sidekick, etc., but requires an extra step to remove the markers.

Yes The block markers are automatically removed (cleared) following a block copy or move.

Horizontal scroll margin
(0=Off, 40 - 254)
[Default = 0]

Determines whether very long lines simply extend past the right side of the screen (and are accessed via horizontal scrolling) or are wrapped onto multiple screen lines (called “continuation lines”).

A value of “0” disables the screen wrapping entirely. Any other value (which must be between 40 and 254) enables the “scroll margin”. Text lines longer than this “scroll margin” are wrapped to the next screen line. When enabled, the first screen column is reserved for the “continuation character”. VEDIT can write the screen faster when the scroll margin is disabled (0).

Horizontal scroll
increment (1 - 100)
[Default = 20]

Determines by how many columns **SCROLL RIGHT** and **SCROLL LEFT** scroll the screen sideways.

Retab during columnar
block insert [Default =
Yes]

Controls whether sequences of spaces in a columnar block insertion are converted to Tab characters. When enabled, the spaces in the inserted text and surrounding spaces in the existing text are converted to Tab characters. This switch does not affect normal block insertions.

During a columnar block copy/ move to a Text Register, Tab characters are always converted to spaces.

Trim spaces after
columnar block insert
[Default = Yes]

Controls whether the trailing spaces of a columnar block insertion at the end of lines are removed. This only trims spaces on those lines involved in the insertion.

Advance cursor past
block insert

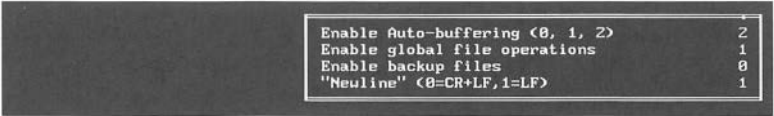
Determines whether the cursor lands at the beginning or end of a block insertion.

No The cursor is not moved, and is left at the beginning of the inserted text.

Yes The cursor is advanced past the inserted text.

File Handling (Submenu)

Controls file related functions.



- Enable backup files
[Default = No]

Controls whether backup files are created, i.e. whether the original file is renamed to “filename.b”.
- "Newline" character(s)
(0=CR+LF, 1=LF) [0 for DOS, 1 for QNX or UNIX]

Determines how VEDIT assumes lines normally end, i.e. what the **newline** character is. Controls screen display and operation of the <Enter> key.

0

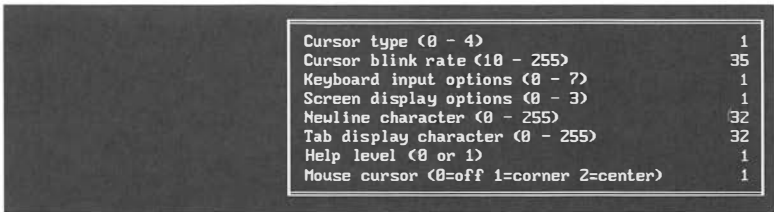
Lines end in Carriage-Return and Line-Feed. Typical for *DOS*. Lines ending in only a Line-Feed (without a preceding Carriage-return) are displayed with a “<LF>”. The <Enter> key inserts both a Carriage-Return and a Line-Feed.

1

Lines end in just a Line-feed. Typical for QNX and for UNIX. Carriage return is no longer a special character and if it occurs in the text it is displayed as “<CR>”. The <Enter> key inserts only a Line-Feed.

Misc (Submenu)

Settings for keyboard input and screen display options.



- Cursor type (0 - 4)
[Default = 1]

Determines how the cursor is displayed.

0

Blinking Underline character.

1

Blinking Reverse Video Block - variable blink rate.

2

Solid Reverse Video Block.

	3	Attribute (uses IBM PC “blink” attribute).
	4	System cursor (Console only).
Cursor blink rate (10 - 255) [Default = 35]		Determines the cursor’s blink rate for cursor types 0 and 1 above. A smaller number causes the cursor to blink faster.
Keyboard input options (0 - 7) [Default = 1]		Controls processing of keyboard characters. You will only need to change this value for special applications.
		This option combines three options into one by having you add “mask” values (setting bits) for each desired sub-option.
		The base value of “0” strips 8 bit characters, discards unassigned control keys and preserves the case of letters.
	Mask 1	Enable 8 bit keyboard characters; you always want this on the console.
	Mask 2	Enter unassigned function/control keys into the text.
	Mask 4	Reverse the case of all letters, e.g. typing “a” gives you “A” and typing “A” gives you “a”. (We are not sure why anyone would want this, but VEDIT has always had this capability.)
		For example, selecting all sub-options would give you a value of “7”.
Screen display options (0 - 3) [Default = 1]		Controls how control and graphic characters are displayed. You will only need to change this value for special applications.
		This option combines three options into one by having you add “mask” values (setting bits) for each desired sub-option.
		The base value of “0” displays control characters (except <tab>, <CR> and <LF>) in the normal “^letter” notation, and graphics characters in the numeric notation “<nnn>”.
	Mask 1	Displays graphics characters literally. For example, on the console foreign letters and box drawing characters are displayed.
	Mask 2	Displays control characters literally. For example, on the console <Ctrl>-<A> is displayed as the “smiling face” instead of as “^A”.



The above two options may be risky on some terminals. They cause VEDIT to simply pass unusual characters to the display hardware. If the hardware does not handle those characters correctly, the effects may be unpleasant.

"Newline" display (0 - 255) [Default = 32]

Determines the character displayed at the end of each line where the **newline** is. The default "space" (value 32) is, of course, not visible. Visible candidates on the console include values "14", "17" and "20".

Tab-display character (0 - 255) [Default = 32]

Determines the "fill" character used to display tab characters on the screen. By default, "spaces" (value 32) are displayed to the next tab position. If you need a better indication of where tab characters are, pick another character such as a "period" (value 46) or value "4".

Help level [Default = 1]

When set to "1", a help line is displayed on the bottom of the screen whenever you are in the menu system or using "Point and Shoot" file selection. "0" turns this help line off.

Mouse cursor (0=Off, 1=Corner, 2=Center) [Default = 2]

Enables mouse support in VEDIT and determines whether the mouse cursor starts up in the center of the screen or "invisibly" in the upper right hand corner. "0" turns off VEDIT's mouse support. Experienced users will usually prefer a value of "1".

Save to Disk

Save the newly configured VEDIT on disk.

Keystroke Equivalent:

(none)

Description:

Any configuration changes you make while in VEDIT and any Keystroke Macros you add are temporary and will be lost when you exit VEDIT unless you explicitly make them permanent with **{CONFIG, Save to disk}**.

To make configuration changes permanent:

1. Select **{CONFIG, Save to disk}**.

You are prompted with:

Save Keystroke Macros too (Y/N)?

2. If you also want to save any new Keystroke Macros you may have added type "Y"; otherwise type "N". You are prompted with:

Enter Filename: /user/fred/.veditrc

A default filename in your home directory is always shown. If VEDIT has been invoked as **vedit** the default filename will be **.veditrc** as shown above.

Otherwise the default filename will be include the name of the editor VEDIT is emulating. For example, if VEDIT has been invoked as **vi**, the default filename will be **.vedit.virc**. This is the initialization file VEDIT reads for that emulation mode.

3. If the default VEDIT filename is correct press <Enter>. Otherwise edit the path and filename as needed.

A complete VEDIT configuration consists of three files:

- a macro language command file such as **/usr/lib/vedit/veditrc**
- a configuration parameter file such as **/usr/lib/vedit/vedit.cfg**
- a keyboard layout file such as **/usr/lib/vedit/vedit.key**

When you save your configuration, two or three (the keyboard layout is optional) files are created. In the typical case, the save creates **\$HOME/.veditrc**, **\$HOME/.vedit.cfg** and **\$HOME/.vedit.key**. If you were emulating **vi**, it would create **\$HOME/.vedit.virc**, **\$HOME/.vedit.vicfg** and **\$HOME/.vedit.vikey**.

These newly created files contain complete VEDIT configuration information. Often there is a good deal of redundancy between these files and the ones in **/usr/lib/vedit**. You may have changed only a few items, but the files repeat everything. This may prevent you from seeing the effects of any improvements the vendor or your system administrator may make in the main files.

There are two solutions. The preferred one is to edit your saved versions to remove the redundancy. The QNX **diff** command (see *Utilities Reference*) or VEDIT **compare** macro (see Chapter 4) may be helpful.

An alternative is to move the created files to a convenient directory (perhaps \$HOME/vedit) and set the variable **VEDIT_PATH** in your QNX environment to name that directory. VEDIT will then look in that directory for its initialization files and come up as you have configured it. Note that it may also be necessary to copy macro files from **/usr/lib/vedit** into your VEDIT directory.

See Also: **{MISC, Keyboard Layout, Load from Disk},**
{MISC, Keyboard Layout, Save to Disk}

Read from Disk

Read configuration information from disk.

Keystroke Equivalent: (none)

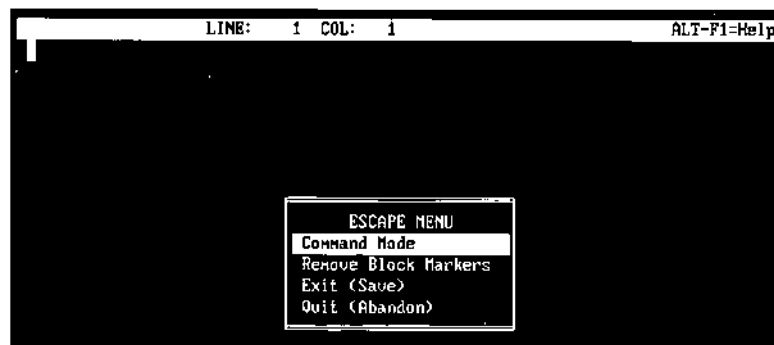
Description: This loads configuration files saved by **{CONFIG, Save to disk}** and changes VEDIT to that configuration.

Default filenames are as for **{CONFIG, Save to disk}**.

Escape Menu

Keystroke Equivalent: <Esc>, This is the Edit Function **ESCAPE**

Introduction: The Escape menu provides quick access to some commonly used functions. These items have been selected because they all represent a conventional and logical use of the <Esc> key, meaning they let you exit, or “escape from” a particular common situation.



To select the Escape Menu, press **ESCAPE**; this is usually the <Esc> key. However, on terminals you may have to press the <Esc> key twice.

Since this key is also used to exit, or “escape from” menu system and other prompts, it will only bring up the Escape Menu when it’s not obvious from what you would like to escape. Of course, you can also escape from the Escape Menu by pressing **ESCAPE**.

Since each of these items have been thoroughly described earlier in this Chapter, we will describe them briefly and list where you can find a full description.

Remove Block Markers

Keystroke Equivalent:

<Ctrl>-<C>, **CANCEL**

Description:

Remove any block markers which are currently set and remove any highlighting.

See Also:

Identical to **{BLOCK, Remove Markers}**.

Exit (Save)

Keystroke Equivalent:

<Alt>-<X>, This is a Keystroke Macro.

Description:

Save the current file and exit VEDIT.

See Also:

Identical to **{FILE, Exit}**.

Quit (Abandon)

Keystroke Equivalent:

(none)

Description:

Quit the current file(s) and optionally exit VEDIT.

See Also:

Identical to **{FILE, Quit}**.



Chapter 6

Edit Function Reference

Edit Functions are the basic building blocks of VEDIT's operation. When you work with VEDIT you press keys which represent either displayable characters or which activate these Edit Functions.

Each Edit Function is assigned to a specific key or key combination. When you press that key or combination you invoke the desired Edit Function. These assignments can be changed to best suit your tastes and needs. For more information on assigning keys to Edit Functions, see {**MISC, Keystroke Macros**} in Chapter 5 (Menu Reference).

You can display which key or keys are assigned to each Edit Function by pressing **HELP** or selecting {**HELP, General**}.

BACKSPACE

Deletes the character to the left of the cursor.

It can be used in dialog boxes to edit the current entry.

BLOCK COPY BLOCK MOVE

Copies or moves a block of text within the current file. First, move the cursor to the beginning of the block and select {**BLOCK, Set Marker**} or press **BLOCK COPY** to set the first block marker. Next, move the cursor to highlight the desired block and press the key again to set the second block marker. Last, move the cursor to the destination for the block and press **BLOCK COPY** to copy or **BLOCK MOVE** to move.

BLOCK COPY performs a convenient combination of the {**BLOCK, Set Marker**} and {**BLOCK, Copy to Cursor**} functions depending upon how many block markers are set:

- If neither block marker is set: Set the first marker.
- If the first marker is set: Set the second marker.
- If both markers are set: Copy the block to the cursor location.

Therefore, **BLOCK COPY** can also be used to mark a block of text for other operations, such as printing using {**PRINT, Block**}, and copying to a Text Register using **T-REG COPY**.

See also: “Block Operations” in Chapter 3 (Editing Guide); the {**BLOCK**} menu in Chapter 5 (Menu Reference).

CANCEL

Cancels and removes the menu system or any prompt. Also stops any **REPEAT** operation. During normal editing (no prompts) it removes any block markers that are set.

See also: “Using Menus” in Chapter 3 (Editing Guide); {**BLOCK, Remove Markers**} in Chapter 5 (Menu Reference).

CURSOR UP

Moves the cursor up one line, to the same horizontal position if possible. If the position is in the middle of a tab character, it moves to the end of the tab. The setting for {**CONFIG, Emulation, Cursor positioning mode**} determines whether the cursor can be positioned past shorter lines.

In dialog boxes, it recalls previous entries.

CURSOR DOWN

Moves the cursor down one line, to the same horizontal position if possible. The commentary for **CURSOR UP** apply.

CURSOR RIGHT

Moves the cursor to the next character. At the end of the line, it moves to the beginning of the next line. Its behavior at the end of a line is changeable with {**CONFIG, Emulation, Cursor positioning mode**}.

In dialog boxes, it permits editing the current entry.

CURSOR LEFT

Moves the cursor to the previous character. At the beginning of a line, it moves to the end of the previous line.

DELETE Deletes the character at the cursor including the **newline** character. Deletes a “Carriage Return” and “Line Feed” pair (**newline**) as a single character.

DEL PREV WORD Delete the word, or portion of a word, to the left of the cursor. When deleting an entire word, it also deletes the space between words.

DEL NEXT WORD Delete the word, or portion of a word, to the right of the cursor. When deleting an entire word, it also deletes the space between words.

ENTER CTRL Enters the next keystroke literally into the text including control and graphics characters. Also used to enter control characters into search/replace strings.

See also: “Keyboard Characters and Screen Display” in Chapter 3 (Editing Guide); {**EDIT, Enter CTRL Char**} and {**MISC, ASCII Table**} in Chapter 5 (Menu Reference).

ERASE EOL Erase (delete) all text from the cursor to the end of the current line.

See also: {**EDIT, Erase EOL**} in Chapter 5 (Menu Reference).

ERASE LINE Erase (delete) the entire current line.

See also: {**EDIT, Erase Line**} in Chapter 5 (Menu Reference).

ESCAPE “Escape” from the current prompt or menu level. If there is no prompt or menu it pops up the “ESCAPE” Menu from which you can cancel operations or exit/quit the editor.

See also: “Using Menus” in Chapter 3 (Editing Guide); “Escape Menu” in Chapter 5 (Menu Reference).

FIND Performs a search operation. Prompts for the “search string” which may be up to 80 characters long; or press **FIND** again to select from the “search options”. Press <Ctrl>-<N> to search for a **newline** (<CR><LF>). Precede control characters you want to search for with **ENTER CTRL**.

Use [CURSOR UP] to recall previous search strings. Press **FIND AGAIN** to search for the next occurrence.

See also: {**SEARCH, Search**} in Chapter 5 (Menu Reference).

FIND AGAIN Repeats the previous **FIND** or **REPLACE**.

See also: {**SEARCH, Again**} and the “[A]gain” search option in Chapter 5 (Menu Reference).

HELP Displays context sensitive help relevant to the current menu or prompt.

While editing, it first displays the keyboard layout.

- For help on a particular edit function press the corresponding key.
- For the menu of help topics, press <Enter> twice. Then type the name of the topic for which you want help.

When the menu system or a prompt is on the screen, **HELP** pops up specific information on the current function.

See also: “On-line” in Chapter 3 (Editing Guide); {**HELP, General**} in Chapter 5 (Menu Reference).

INDENT Increases the left margin by the amount of the “Indent Increment”. Each new line is then automatically padded with tabs and spaces to the left margin. If the cursor is before any text on the current line, pads the current line to the new left margin.

If the cursor is within a highlighted block (both block markers must be set) it increases the indentation of all lines within the block.

See also: “Indenting Text (Left Margin)” in Chapter 3 (Editing Guide); {**CONFIG, Word Processing, Left Margin**} and {**EDIT, Format Paragraph**} in Chapter 5 (Menu Reference).

INSERT TOGGLE Toggles between “Insert” and “Overstrike” modes.

See also: {**EDIT, Insert Mode**} in Chapter 5 (Menu Reference).

LINE BEGIN Moves the cursor to the first character of the screen line or, if already there, to the first character of the previous screen line. The behavior of this function is changeable using {**CONFIG, Emulation, Line emulation mode**}.

LINE END	Moves the cursor to the end of the current screen line. If already there, it moves the cursor to the end of the next screen line. The behavior of this function is changeable using {CONFIG, Emulation, Line emulation mode} .
MENU	Starts the pull-down menu system. The menu system is accessed in the usual manner using the cursor keys and <Enter>. Pressing ESCAPE backs out of the menu system, one level at a time. Pressing CANCEL cancels any prompts and removes the menu system. See also: “Using Menus” in Chapter 3 (Editing Guide).
NEXT LINE	Moves the cursor to the beginning of the next text line.
NEXT PARAGRAPH	Moves the cursor to the beginning of the next paragraph.
NEXT TAB STOP	Moves the cursor to the next tab stop.
See also:	{CONFIG, Tab Stops} in Chapter 5 (Menu Reference).
NEXT WORD	Move the cursor to the beginning of the next word.
PAGE UP	Moves the cursor to the previous screen “page”—similar to pressing CURSOR UP for 3/4 screen lines.
PAGE DOWN	Moves the cursor to the next screen “page” — similar to pressing CURSOR DOWN for 3/4 screen lines.
PREVIOUS PARAGRAPH	Moves the cursor to the beginning of the current paragraph. If already at the beginning of a paragraph, moves the cursor to the beginning of the previous paragraph.
PREVIOUS WORD	Moves the cursor to the beginning of the current word. If already at the beginning of a word, moves the cursor to the beginning of the previous word.
REPEAT	Repeats the next typed character, Edit Function, or menu selection. The repeat count is either a multiple of four or the number typed in. Type “000” and any character to abort; or press CANCEL to abort. See also: {EDIT, Repeat} in Chapter 5 (Menu Reference).

REPLACE	Performs selective or global search and replace. Prompts for the “search string” and then for the “replacement string”. After locating the text, you have the option of replacing it or not and immediately searching for the next occurrence, of globally replacing all occurrences, of replacing only one occurrence, or or cancelling the command. After the command is done, you can press FIND AGAIN to search for the next occurrence and have the same options.
See also:	[FIND] for details about the search string and options; {SEARCH, Replace} in Chapter 5 (Menu Reference).
RETURN	Is always assigned to the <Enter> key. In Overstrike Mode it moves the cursor to the next line, same as NEXT LINE . In Insert Mode or at the end of the file it opens up a new line by inserting a newline character. Pressed in the middle of a line, it splits the line. Its behavior can be changed with {CONFIG, Emulation, Typewrite emulation mode} .
SCREEN BEGIN	Moves the cursor to the beginning of a line at or near the top of the current window (screen). The behavior of this function is changeable with {CONFIG, Emulation, Screen emulation mode} .
SCREEN END	Moves the cursor to the end of a line at or near the end of the current window (screen). The behavior of this function is changeable with {CONFIG, Emulation, Screen emulation mode} .
SCROLL UP	Scrolls the screen to show the next previous line at the top of the screen. Moves the cursor, if required, to keep it on-screen.
See also:	“Scrolling the Screen” in Chapter 3 (Editing Guide).
SCROLL DOWN	Scrolls the screen to show the next line at the bottom of the screen. Moves the cursor, if required, to keep it on-screen.
SCROLL RIGHT	Scrolls the screen to view long lines going off the right side of the screen. Moves the cursor, if required, to keep it on-screen. It scrolls by the number of columns defined by {CONFIG, Emulation, Horizontal scroll increment} .
SCROLL LEFT	Scroll the screen to view the beginning portions of long lines. Moves the cursor, if required, to keep it on-screen.

TAB CHARACTER

- In Insert Mode or at the end of a line, it inserts a Tab character (or optionally spaces to the next tab position) into the text.
- In Overstrike Mode, it moves the cursor to the next tab position, similar to **TAB CURSOR**.

Its behavior is changeable with
{**CONFIG, Emulation, Typewriter emulation mode**}.

See also: “The <tab> Key and Tab Characters” in Chapter 3 (Editing Guide); {**CONFIG, Emulation, Expand <tab> key with spaces**} in Chapter 5 (Menu Reference).

T-REG COPY T-REG MOVE

Lets you copy or move blocks of text to the Text Registers. First, move the cursor to the beginning of the block and select {**BLOCK, Set Marker**} to set the first block marker. Next, move the cursor to highlight the desired block. Then press **T-REG COPY** to copy the block to a Text Register, or **T-REG MOVE** to move the block. You are then prompted for the desired Text Register’s single character name.

Like most other block functions, if no block marker is set, these functions simply set the first block marker.

See also: “Block Operations - The Text Registers” in Chapter 3 (Editing Guide); {**BLOCK, Copy to Register**} in Chapter 5 (Menu Reference); **BLOCK COPY** above.

T-REG INSERT

Inserts the desired Text Register at the current cursor position. You are again prompted for the Text Register’s name.

You can insert a Text Register after you have started to highlight a block. If the block you copied/moved to the Text Register was a columnar block, it will also be inserted as a columnar block.

See also: “Block Operations - The Text Registers” in Chapter 3 (Editing Guide); {**BLOCK, Insert Register**} in Chapter 5 (Menu Reference).

UNIDENT

Decrease the left margin by the “Indent Increment”. If the cursor is before any text on the current line, moves the line to the new left margin.

VISUAL ESCAPE

See also: **INDENT**; {**CONFIG**, Word Processing, Left Margin} and {**EDIT**, **Format Paragraph**} in Chapter 5 (Menu Reference).

VISUAL ESCAPE

This edit function exits the “Visual Mode” and aborts any command macro, such as **wildfile.vdm**, which is currently running.

The effect of this function is identical to {**ESCAPE**, **Command Mode**}.

VISUAL EXIT

This edit function takes you out of the "Visual Mode" normally used in editing and into VEDIT "Command Mode".

It is used in Keystroke Macros to access the VEDIT macro language. For example, the Keystroke Macro **VISUAL EXIT] OK** switches to Command mode and gives the macro language command "OK" which deletes text from the cursor to the beginning of the line.

It also permits exiting the “Visual Mode” so that a command macro, such as **wildfile.vdm**, can continue running.

It enters the “Command Mode” when no macro is currently running.

Chapter 7

Configuration

For the most part, you can configure VEDIT from within itself and then save the new configuration using {CONFIG, Save to disk}.

However some configuration changes can only be made by editing the `.cfg` files. The format of these files is described in this chapter.



Creating a new VEDIT configuration solely by editing `.cfg` files is possible but *not* recommended. Instead, *do as much as possible from within* VEDIT, interactively testing your changes as you go. Save that configuration, then edit the resulting `.cfg` file as necessary.

Location of Configuration Files

The `/usr/lib/vedit` directory comes with the configuration file `vedit.cfg` for VEDIT “just being itself” and additional `.cfg` files for the supplied emulations, for example `qed.cfg` and `vi.cfg`. System administrators may edit these files and/or create additional ones to support other emulations.

{CONFIG, Save to Disk} saves the current VEDIT configuration to a user-selected filename. The default is `$HOME/.vedit.cfg` (or `$HOME/.vi.cfg` and `$HOME/.qed.cfg` for the emulations).

Additional `.cfg` may be created in other directories as necessary, either from within VEDIT with {CONFIG, Save to disk} or with QNX commands such as `cp`.

What can be Changed via Configuration Files?

Editing configuration files lets you configure the following parameters:

- Colors (attributes) used for text, the status line, window borders, menus, block highlighting, and the cursor.
- Override the current screen size and have VEDIT start up with a particular size, e.g. always start up in VGA 50 line mode.
- Characters used for drawing boxes around menus and windows.
- Enable or disable the “shell” functions.
- Number of Undo levels available.
- Number of lines of screen overlap when using **PAGE UP** and **PAGE DOWN**.
- The range of screen lines in which the cursor can move before the screen scrolls.
- Whether VEDIT starts up in Command Mode or the normal Visual Mode.
- How many times the Command Mode help message is displayed.
- Anything changeable from within VEDIT via the {**CONFIG**} menu or {**PRINT, Config**} and {**SEARCH, Options**} submenus.

Format of Configuration Files

The configuration files contain macro language commands to set up VEDIT. Nearly all lines of the files are of the form:

command index "comment" value

where the parts of the command line are:

- *command*: a command whose only effect is to load one of VEDIT's internal variables with a *value*.
- *index*: a numeric index telling which variable to load
- *comment*: a comment, ignored by VEDIT
- *value*: the numeric value to load

The only exception to this pattern is the last line which sets tab stops. It has just the *command* **ET** and a series of *values*.

Both the *indexes* and the *values* are decimal numbers.

Caution Users should change *only* the *comment* and *value* portions of these lines and should not alter the order of lines.

Users should not delete lines from the system **.cfg** files in **/usr/lib/vedit**. Deleting lines from user **.cfg** files is OK. VEDIT will simply revert to the settings in the system **.cfg** files.

Because VEDIT is maintained in compatible versions for several environments, a number of parameters not applicable to the QNX version exist in the configuration files. These are shown below, marked “Not applicable on QNX”. *Users should leave these parameters unchanged.*

**Configuration
File, Line by Line**

This section illustrates a configuration file line by line with explanations of what each line does and how you can change it. Lines are shown in the order in which they actually occur in the file.

YJO 1 "Undo levels - max"	500
----------------------------------	------------

This determines the number of levels available on the Undo stack when a newly created edit buffer has enough memory for its own reserved 64K memory segment. (Each edit buffer has its own Undo stack). See Appendix B (Memory Management) for a complete explanation.

YJO 2 "Undo levels - min"	100
----------------------------------	------------

This determines the number of levels available on the Undo stack when a newly created edit buffer has to share memory with other edit buffers or Text Registers.

YJU 1 "Look for VEDIT files in current Directory"	0
--	----------

Not applicable on QNX. *Do not change.*

YJU 2 "Enable auto-startup"	1
------------------------------------	----------

Not applicable on QNX. *Do not change.*

YJU 3 "Enable shell"	1
----------------------	---

Determines whether the functions {**MISC, Shell**} and {**MISC, Run QNX Program**} are available in the menu system and whether related commands are available in command mode.

This parameter is usually set to “Yes”. However, systems integrators can disable these functions when VEDIT is run from inside a tight environment, such as a vertical application in which the users need no access to QNX.

The **-r** command-line option also disables these functions, along with some others. See “Options” in Chapter 3 for details.

YJU 4 "Enable OS/2 operation"	1
-------------------------------	---

Not applicable on QNX. *Do not change.*

YJU 5 "Keyboard input {0=ROM,1=System}"	0
---	---

Not applicable on QNX. *Do not change.*

YJU 6 "Start up in Command Mode"	0
----------------------------------	---

Most users want VEDIT to start up in Visual Mode. However, if you prefer, you can start up in Command Mode.

YJU 7 "Count for Command Mode help message"	3
---	---

This parameter determines how many times the **COMMAND:** prompt is preceded by the “**EX**” to **Exit**, “**EQ**” to **Quit** . . . help message. Experienced users may prefer a value of 0 to disable it. A value of 255 causes it to always appear.

YJU 8 "Speed of computer {Mhz}"	4
---------------------------------	---

Not applicable on QNX. *Do not change.*

Screen Color
Parameters

All the colors used by VEDIT can be configured. The text color can be changed from inside VEDIT, however the other colors can only be changed here.

On a monochrome console, the only attributes are “normal” (any value 1 to 7), “reverse video” (value 112) and “highlighted” (value 15).

On a color console, various colors can be selected. The table below lists the colors available.

Value	Color	Value	Color
16	Black	1	Blue
	on Blue		on Black
32	Green	33	Green
48	Cyan	49	Cyan
64	Red	65	Red
80	Magenta	81	Magenta
96	Brown	97	Brown
112	White	113	White
2	Green	3	Cyan
	on Black		on Black
18	Blue	19	Blue
50	Cyan	35	Green
66	Red	67	Red
82	Magenta	83	Magenta
98	Brown	99	Brown
114	White	115	White
4	Red	5	Magenta
	on Black		on Black
20	Blue	21	Blue
36	Green	37	Green
52	Cyan	53	Cyan
84	Magenta	69	Red
100	Brown	101	Brown
116	White	117	White
6	Brown	7	White
	on Black		on Black
22	Blue	23	Blue
38	Green	39	Green
54	Cyan	55	Cyan
70	Red	71	Red
86	Magenta	87	Magenta
118	White	103	Brown

The following color attribute values may be set:

YJA	1	"Color for edited text"	130
YJA	2	"Color for status line "	240
YJA	3	"Color for status line messages"	240
YJA	4	"Color for Window border "	131
YJA	5	"Color for Window border message"	176
YJA	6	"Color for dialog prompts"	151
YJA	7	"Color for main menu "	240
YJA	8	"Color for main menu selection"	255
YJA	9	"Color for main menu highlighting"	151
YJA	10	"Color for pull-down menu "	199
YJA	11	"Color for pull-down menu selection"	207

You may also set the highlight colors which various actions will use:

YJA	12	"Color override for pull-down menu "	0
YJA	13	"Color override for block highlighting"	0
YJA	14	"Color override for Cursor (0 = auto) "	0

For these "override" colors, a setting of zero means that VEDIT should determine the colors to use based on the existing screen colors.

Special Screen
Characters

On the console VEDIT uses graphics characters to draw boxes around its menus and dialog boxes and to indicate which menu choices lead to submenus.

These characters may be changed by editing the lines below:

YJB	1	"Horizontal border character"	205
YJB	2	"Vertical border character"	179
YJB	3	"Upper-Left corner character"	213
YJB	4	"Upper-Right corner character"	184
YJB	5	"Lower-Left corner character"	212
YJB	6	"Lower-Right corner character"	190
YJB	7	"Top intersect character"	209
YJB	8	"Sub-menu (Right) "	16
YJB	9	"Sub-menu (Left) "	17

Screen Size Parameters

Any screen size up to 70 lines by 250 columns is supported by VEDIT.

Normally, VEDIT adjusts itself to the current size of the screen as reported by QNX. To get this behavior, set the configuration file screen size parameters to zero.

Alternatively, you can have VEDIT attempt to change to a particular screen size as it starts up. For example, with VGA hardware and QNX set up to use it, you could have VEDIT switch to 50 line mode. To get this behavior, set the configuration file screen length parameter to 50.

VEDIT's attempt to change screen size succeeds only if both your hardware and the QNX video library in use support the size requested. Failure is not disastrous; it simply leaves you in VEDIT with the size settings QNX had before VEDIT started.

The configuration file lines in question are:

YJN	1	"Number of screen lines"	25
YJN	2	"Number of screen columns"	80

Other parameters also affect how VEDIT uses the screen:

YJN	3	"Size of screen page"	22
-----	---	-----------------------	----

Determines the number of screen lines that [PAGE UP] and [PAGE DOWN] move by. This controls how much screen overlap there is while paging. About 10% overlap is suggested, e.g. "22" for a 25 line display. Having some overlap provides a visual reference while paging and makes scanning a document much easier.

YJN	4	"Top line for Cursor "	3
-----	---	------------------------	---

This sets the top screen line (not including the status line) that the cursor can normally be on, before the screen will begin to scroll down. Setting this value greater than "1" ensures that you will normally see some lines above your present cursor location.

YJN	5	"Bottom line for Cursor "	22
-----	---	---------------------------	----

This is similar to the previous parameter, except that it sets the bottom line range for the cursor. This number must be greater than or equal to the "Top Line for Cursor" setting, and must be at most one less than the "Number of Screen Lines". "3" less than the number of

screen lines is a good starting point. As with the screen paging parameter, setting this parameter to less than the number of screen lines ensures that some lines below your present cursor location will normally be visible.

This value assumes the screen size set by the "screen lines" parameter, or "25" if screen lines is set to "0=auto". VEDIT adjusts this value according to the size of the current window.

YJN	6	"Status line character"	32
-----	---	-------------------------	----

The "empty" parts of a reverse video status line are usually displayed with the space character (value 32). However, if you do not select a reverse video status line, you may want to select a "dash" (value 45) or value 196.

YJN	7	"Screen continuation character"	173
-----	---	---------------------------------	-----

This is the line continuation indicator which is displayed in the reserved column 1 when long lines are wrapped on the screen. Most common is a hyphen (value 45) or reverse video hyphen (value 173). See "Wrapping Long Lines on the Screen" in Chapter 3 (Editing Guide).

YJN	8	"Restore screen mode on exit"	1
-----	---	-------------------------------	---

Not applicable on QNX. *Do not change.*

Other Screen Parameters

A number of other parameters influence other aspects of screen handling. These are all either inapplicable to QNX or applicable only on the console.

YJM	1	"Write screen via BIOS"	0
-----	---	-------------------------	---

Not applicable on QNX. *Do not change.*

YJM	2	"Cursor character"	95
-----	---	--------------------	----

This is the character used for cursor type "0"—the blinking "underline" cursor. While normally the underline character (value 95), a value of "127" gives a solid block on some displays. On the console, cursor type 0 is not a good choice so this setting is not important.

YJM 3	"Screen erase character"	32
-------	--------------------------	----

VEDIT normally clears the screen with spaces (value 32). However, for special applications, a different character could be used.

YJM 4	"High speed CGA"	1
-------	------------------	---

Not applicable on QNX. *Do not change.*

Printer Parameters

These parameters appear in the `.cfg` file but would not normally be set by editing it.

Instead, change them from within VEDIT using the **{PRINT, Config}** submenu and save the configuration.

Refer to the section on that submenu in Chapter 5, Menu Reference, for details on these parameters.

PP	1	"Paper length (5 - 100) "	66
PP	2	"Top margin (0 - 60) "	3
PP	3	"Bottom margin (0 - 60) "	3
PP	4	"Left margin (0 - 100) "	10
PP	5	"Line spacing (1 - 4) "	1
PP	6	"Enable Form-Feed"	1
PP	7	"Force page eject"	0
PP	8	"Printer {0=/dev/par} "	0

Additional Display and Mode Parameters

A number of parameters also appear in the configuration file which are settable from within VEDIT via the various submenus of the **{CONFIG}** menu or via the **{SEARCH, Options}** submenu.

See Chapter 5 for more information on these. As with the printer configuration options above, the recommended procedure is to change them from within VEDIT rather than by editing the `.cfg` files.

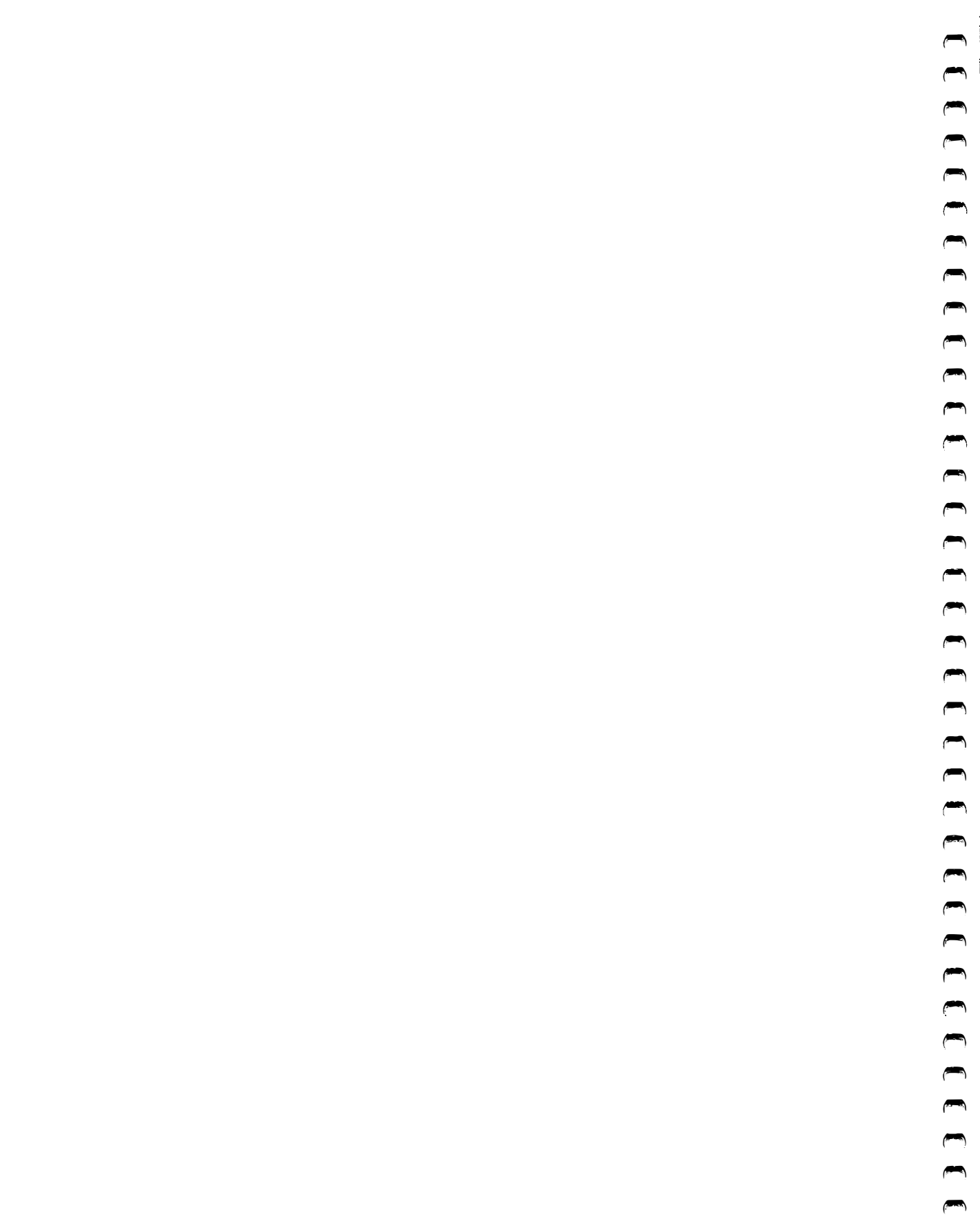
A few parameters appear among this group in the configuration files yet are not accessible from the menus in the QNX version of VEDIT. These are not applicable on QNX and *should not be changed*.

Configuration File, Line by Line

EO	1	"Cursor type (0 - 4)"	1
EO	2	"Cursor blink rate (10 - 255)"	35
EO	3	"Cursor positioning mode (0 - 4)"	1
EO	4	"Typewriter emulation mode (0 - 2)"	2
EO	5	"Expand ab key with spaces"	0
EO	6	"Reserved"	1
EO	7	"Auto-Cancel Block Markers"	1
EO	8	"Line emulation mode (0 - 3)"	3
EO	9	"Reserved"	0
EO	10	"Keyboard input options (0 - 7)"	1
EO	11	"Screen display options (0 - 3)"	1
EO	12	"Delay for Command Mode scrolling (0 - 255)"	0
EO	13	"Emulation (0=VEDIT, 1=vi, 2=QNX)"	0
EO	14	"Help level"	1
EP	1	"User configured parameter"	0
EP	2	"Auto-indent mode"	0
EP	3	"Indent increment (1 - 20)"	4
EP	4	"Lower/Upper case key conversion (0 - 3)"	0
EP	5	"Conditional conversion character (32 - 126)"	59
EP	6	"Left margin (0 - 80)"	0
EP	7	"Right margin (0=Off, 1 - 255)"	0
EP	8	"Newline display character (0 - 255)"	32
EP	9	"Tab display character (0 - 255)"	32
EP	10	"Horizontal scroll margin (0=Off, 40 - 254)"	0
EP	11	"Horizontal scroll increment (1 - 100)"	20
EP	12	"File type (0=Binary, 1=Text, 2, 3)"	0
EP	13	"Newline character(s) (0=CR+LF, 1=LF)"	1
ES	1	"Enable Auto-buffering (0, 1, 2)"	2
ES	2	"Retab after columnar block insert"	1
ES	3	"Trim spaces after columnar block insert"	1
ES	4	"Advance cursor past block insert"	1
ES	5	"Equate Lower/Upper case during search"	1
ES	6	"Regular expression search mode"	1
ES	7	"Enable Undo in Command Mode"	1
ES	8	"Suppress error handling"	0
ES	9	"Explicit string delimiters"	1
ES	10	"Enable global file operations"	1
ES	11	"Justify paragraphs (0=Off, 1=On, 2=Unjustify)"	0
ES	12	"Enable backup files"	0
ES	13	"Insert mode (Visual Mode)"	1
ES	14	"Columnar block mode (Visual Mode)"	0

The last parameter sets tab stops. It is one long line in the file and *should be left as one line after any editing* you choose to do. We show it below split onto multiple lines for printing convenience.

ET	5	9	13	17	21	25	29	33	37	41
	45	49	53	57	61	65	69	73	77	81
	85	89	93	97	101	105	109	113	117	121
	125	129								



Chapter 8

Messages

VEDIT displays a message to notify you of errors or special conditions.

Many are simply confirmation prompts to perform an operation which may be hard to undo. Most error messages result when you mistakenly attempt an impossible operation, such as loading a non-existent file.

Messages can also result when VEDIT detects syntax and programming errors in macros written in the VEDIT macro language. In most cases, the offending macro command is displayed.

ABANDON FILE (Y/N)?

This is the normal confirmation prompt for the **{FILE, Quit}** and **{ESCAPE, Quit}** functions. Type “Y” or “y” if you really want to abandon the file without saving any changes.

ABANDON ALL FILES (Y/N)?

This is the confirmation prompt to quit/abandon ALL edit buffers and exit VEDIT.

BAD FILENAME

The specified filename does not follow QNX conventions. Perhaps the pathname or either part of the filename is too long.

BAD PARAMETER

You specified an invalid command parameter. With **{WINDOW, Create}** you attempted to create a window which already exists or you specified too small/large a size. Can also result from errors related to the VEDIT macro language.

BREAK

Typically this indicates that you pressed **CANCEL** (<Ctrl>-<C>) to interrupt a VEDIT operation, e.g. to stop printing or to abort a search through a large file when you realize the search string is ill-chosen or mistyped.

It also appears for any condition which sends a break signal (see the **signal** section of the *Programmer's Guide*) to the VEDIT process. Such conditions include:

- Pressing the <Ctrl>-<Break> key combination within VEDIT.
- Using the **kill** or **slay** utilities to signal VEDIT.
- VEDIT signalling itself when it encounters something it cannot handle. For example, you attempted to edit a 60K file consisting of just one long line.

CANNOT ENTER VISUAL MODE

VEDIT noticed that its internal keyboard layout table has become corrupted; it will immediately exit.

CANNOT FIND: *search string*

The specified search string could not be found. This is a normal message for some macros written in the VEDIT macro language.

CANNOT OPEN TWO FILES

A macro written in the VEDIT macro language attempted to open two files in one edit buffer.

CANNOT RUN SHELL

Usually indicates that there is not enough memory to start a new shell. Perhaps you can exit one or more edit buffers to free more memory space.

The same error message occurs if QNX cannot find the shell executable or if you lack permission to execute it.

CLOSE ERROR

The output file could not be closed and therefore is not yet saved! This is an unusual condition, generally indicating outside interference between VEDIT when opened the file and when it tried to close it—for example a floppy disk has been removed or a network node powered off during your edit session.

Use **{FILE, Save as}** to save the file elsewhere.

COMMAND NOT AVAILABLE IN BROWSE MODE

Some commands are not available when you have invoked VEDIT with **-b** for the “browse” mode. These include **{BLOCK, Write to Disk}** and **{FILE, Save As}**.

DIRECTORY NOT FOUND

A directory specified as part of a file pathname could not be found. Typographical errors are much the commonest cause of this error message. Other possibilities: the directory does not exist or you lack permission to access it.

END OF BUFFER REACHED

Caused by a programming error within a macro written in the VEDIT macro language.

END OF MACRO REACHED, STRING STILL OPEN

Caused by a programming error within a macro written in the VEDIT macro language.

END OF MACRO REACHED, STRUCTURE STILL OPEN

Caused by a programming error within a macro written in the VEDIT macro language.

FILE ALREADY BEING EDITED

The file you are trying to edit in the current edit buffer is already being edited in another edit buffer. You cannot edit the same file in two edit buffers. Use **{HELP, Edit Buffers}** to check this. On a network system it is possible that another user is already editing the desired file.

FILE IS R/O

You cannot alter a file which is “read-only” (R/O). All files are “read-only” if you invoke VEDIT with the **-b** (browse mode) option. A file for which you do not have QNX write permission (see *Utilities Reference* for **chmod**) is automatically marked “read-only” by VEDIT. Additional files may be marked “read-only” by the user with **{FILE, R/O File}**.

FILE NOT FOUND

The file you specified for editing does not exist. The cause is often a typo.

FILE NOT OPENED

This message indicates that VEDIT could not open the file for read or write, whichever was required. It usually indicates a problem with QNX access permissions—see *Utilities Reference* for **chmod**.

FILE POSITIONS NOT IN MEMORY

The **{BLOCK, Copy to Cursor}** and **{BLOCK, Move to Cursor}** functions require that the text block be in memory. You will need to use a Text Register to copy or move blocks between distant parts of a large file.

INVALID COMMAND

Caused by a programming error within a macro written in the VEDIT macro language.

INVALID DRIVE

The drive you specified as part of a filename was either specified incorrectly or does not exist. Most likely you mistyped it.

INVALID EDIT BUFFER OPERATION

You are attempting an operation which is valid for normal Text Registers, but is invalid for edit buffers. You cannot change the contents of an edit buffer except when it is the active edit buffer. Usually caused by {BLOCK, Copy to Register}.

INVALID [HELP] REQUEST

During the [HELP] function you are prompted to press the function/control key for the desired function. Instead you pressed a displayable character or an unused function/control key.

INVALID INTERNAL VALUE

Caused by a programming error within a macro written in the VEDIT macro language.

INVALID NUMERIC REGISTER

Caused by a programming error within a macro written in the VEDIT macro language.

INVALID SEQUENCE

While setting up a keystroke macro you entered the “Function/Control Key” as an invalid “sequence” beginning with a displayable character. You may use a function/control key, a single displayable character or a “sequence” beginning with a control or function key.

INVALID TEXT MARKER

Caused by a programming error within a macro written in the VEDIT macro language.

JP COMMAND SYNTAX ERROR

Caused by a programming error within a macro written in the VEDIT macro language.

JUMPING INTO A FLOW CONTROL STRUCTURE

Caused by a programming error within a macro written in the VEDIT macro language.

LABEL MISSING: *!label!*

Caused by a programming error within a macro written in the VEDIT macro language.

MACRO ERROR IN *r*

Caused by a programming error within a macro written in the VEDIT macro language.

NESTING (STACK) ERROR

Caused by a programming error within a macro written in the VEDIT macro language.

NEW FILE

The file specified for editing did not exist on disk and a new file has been created. If you typed the wrong filename, you can edit the correct file by immediately selecting **{FILE, New File}**.

NO CURRENT SEARCH/REPLACE STRING

The **FIND AGAIN** function did not operate because you have not yet specified a search or replace string with **FIND** or **REPLACE**.

NO DISK SPACE

The disk became full before the entire file(s) was saved. To save your file(s) you must either first delete some unneeded files (switch QNX consoles or use VEDIT's **{MISC, Shell}** to do this) , or save the file to another disk using **{FILE, Save As}**.

See "Maximum File Size" in Appendix A for a description of how much disk space VEDIT requires to edit files.

NOTE:



Never delete any files from within VEDIT which have "\$" in their filename extension, e.g. ".@\$ \$" and ".@R\$ ". They are temporary file needed by VEDIT. You can delete them after exiting VEDIT in the unlikely event they still exist.

NO OUTPUT FILE

There is no (output) filename specified into which to save the current edit buffer, e.g. with **{FILE, Exit}**.

If you have already saved the file to disk and there is nothing further to save, you can exit with **{FILE, Quit}**.

To save the text select a filename with **{FILE, Save As}**.

NOT ENOUGH MEMORY FOR OPERATION

There was insufficient free memory to perform the operation, such as copying a block of text to a Text Register. Other than some block operations, this error will rarely occur as long as **{CONFIG, File Handling, Enable Auto-buffering}** is set to "2".

NOT ENOUGH MEMORY TO AUTO-BUFFER

There is insufficient free memory in the edit buffer to perform auto-buffering. Most likely you are simultaneously editing more files than VEDIT can handle with the available memory. Use **{HELP, Status Display}** to see how much memory space is free. Most likely, almost the entire memory is being used by other edit

buffers and Text Registers leaving less than 3 Kbytes for the current edit buffer. You must empty some Text Registers and/or finish up editing other files.

NOT ENOUGH MEMORY TO LOAD VEDIT

There is insufficient memory available for VEDIT to start up. VEDIT requires enough memory to load one copy of the program itself per node, plus about 40K per user as a minimum data space. Use the QNX **sin info** command to see how much is available.

NOT FOUND IN HELP FILE

The help topic you selected could not be found in the help file. Either you entered the topic name incorrectly or the help file is not complete.

OK TO DELETE *nnnnn* LINES AND RESET UNDO (Y/N)?

There is insufficient free memory to delete the (large) block of text *and* let you Undo it if needed. To delete this block it will also be necessary to reset the Undo stacks. This prompt does not occur if the deletion can immediately be undone.

OK TO ERASE THESE FILES (Y/N)?

This message only occurs when a macro written in the VEDIT macro language uses the “EK” (Kill file) command to delete files from disk. If you type “Y” or “y”, the listed files will be deleted from disk.

OK TO OVERWRITE EXISTING FILE (Y/N)?

This prompt appears with the **{BLOCK, Write to Disk}** and **{MISC, Keyboard Layout, Save to Disk}** functions if the designated file to be written already exists on disk. Type “Y” or “y” to overwrite the existing file. Note that these functions DO NOT create a “backup” of any existing file.

PRINTER/DEVICE NOT READY

Your printer does not respond—most likely it is not turned on or is not set “online”.

This message should never occur if you are using default settings, sending printable output to the QNX spooler which should never be off-line.

The message can occur if you are outputting to an actual device.

Press “R” to retry after setting your printer online. Press “C” to cancel the print command.

PRINTING -- Press <Ctrl>-<C> to Abort

This message is displayed anytime text is being printed. It reminds you that you can press <Ctrl>-<C> to stop the printing.



Pressing <Ctrl>-<C> *only* stops output from VEDIT. Text already given to a spooling program or already stored in a printer’s buffer will continue printing. *To actually stop printing*, you need to action further down the chain—for example, take the printer offline or issue commands to the spool program.

REDEFINE DISPLAYABLE CHAR (Y/N)?

The “Function/Control Key” you entered while adding a new keystroke macro is a displayable character which is about to be assigned a new meaning. Type “Y” to redefine it, or “N” if you made a mistake and do not want to redefine it.

REDEFINE EXISTING KEY (Y/N)?

The “Function/Control Key” you entered while adding a new keystroke macro is already assigned to an edit function or a keystroke macro. Type “Y” to redefine it, or “N” if you made a mistake and do not want to redefine it.

REGULAR EXPRESSION SYNTAX ERROR [SEARCH]

There is a syntax error on the search side of a regular expression. Perhaps a], [{ or } character is missing or the groups are not nested properly. The *or* operator | cannot occur within groups - only between groups. A regular expression cannot begin with *. When using “\n”, be sure that the 'n'th group already exists.

REGULAR EXPRESSION SYNTAX ERROR [REPLACE]

There is a syntax error on the replacement side of a regular expression. Most likely you are improperly using “\n”—be sure that the 'n'th group was defined on the search side. (Counting starts at 1.)

REGULAR EXPRESSION - NOT ENOUGH MEMORY

There is insufficient free memory available to perform the search and replace. Most likely due to an “\n” on the replacement side which corresponds to many (thousands of) characters matched on the search side. Can also result from an improperly formed expression.

REV FILE OPEN

Caused by a programming error within a macro written in the VEDIT macro language.

TOO MANY FILES

Either VEDIT is trying to open more files than any one process is allowed to have or the QNX file system, handling files for many processes, has hit its limit. Close some files or consult your system administrator.

WAITING FOR PRINTER -- Press <Ctrl>-<C> to Abort

VEDIT is waiting on a network or multi-user system for another program or user to release the printer before it can begin printing. This message changes to the normal printing message once printing begins.

WRITE ERROR

An error occurred writing to disk. See below.

READ ERROR

An error occurred reading from disk.

On a floppy disk probable causes include disk not inserted, drive door not closed, using an unformatted disk or using the wrong type of disk. It is also possible the diskette is damaged—replace it and try again.

For a disk being accessed across the network, these messages may indicate that the remote machine has been turned off or rebooted.

Any occurrence of these errors not related to the obvious causes above should be reported to your system administrator.



Appendices

File Management

This is a technical description of VEDIT file handling. It explains the automatic file buffering used to handle large files. For most applications, it is not necessary to have a detailed knowledge of how VEDIT manages large files and memory.

Basic File Handling

The purpose of editing is to either create a new file, or to modify an existing file. When a file is first created, the initial text is entered with the editor, corrections are made, and the text is then saved on disk. When an existing file is edited, it is first read from disk, modified, and then written back to disk with either the original filename or a new filename.

Since VEDIT can edit files which are much larger than memory, it processes two files while editing—it reads text from the existing file called the *input* file and writes text to a new file called the *output* file.

When VEDIT edits an existing file, it performs the following operations:

1. The existing file is opened as the “input file”. As much of it as will fit is read into memory. If the file is small, all of it will be read into memory.
2. A new file is created for the “output file”. Its filename is composed of four parts, separated by periods:

- the original filename. Normally, all of this name is preserved, but long names may be truncated on the right to allow the complete output file filename to fit within the QNX 4.0 restriction of 48 characters for total filename length.
- node number on QNX network (up to 10 decimal digits).
- QNX process ID of VEDIT process (up to five decimal digits).
- a three-character string such as @\$\$ or AR\$. The first character is an edit buffer identifier: @ for the main buffer, a single letter or digit for other buffers. The second character indicates file type: \$ for normal VEDIT output files, R for files created temporarily during backward movement in files. The third character is always \$; look for it if you need to recognize VEDIT temporary files.

These components are intended to guarantee unique filenames. Even with several machines each running several instances of VEDIT, each with several buffers looking at the same file, this guarantees unique filenames.

3. The file is edited as desired. In the case of large files, VEDIT will shuffle text between the input and output files so that the desired portion is in memory and can be displayed on the screen.
4. When saving the file, VEDIT renames the output file to the real name of the destination file.

If automatic backups are enabled, VEDIT also renames the existing input file, adding an extension of “.b”. This is referred to as the “*backup*” of the file. Any previous backup of the file is deleted by this process.

Automatic File Buffering

When editing files which are larger than can fit into memory at one time, VEDIT shuffles text between the input and output files so that the desired portion is in memory and can be displayed on the screen. We call this shuffling “*auto-buffering*”.

It helps to consider the displayed screen a “window” into the edit buffer. This “window” may readily be moved anywhere within the edit buffer with the **PAGE UP**, **PAGE DOWN** and other cursor movement functions. Furthermore, the edit buffer can be considered a “window” into a large file. Moving this edit buffer “window”

toward the end of the file is referred to as “*forward file buffering*”, and moving it toward the beginning of the file as “*backward file buffering*”.

VEDIT also performs auto-buffering, when necessary, to insert large blocks of text. For example, if there is not enough free memory in the edit buffer to insert the contents of a large Text Register, part of the edit buffer will be written to disk to make the necessary memory free.

Backward File Buffering

When editing a large file you often want to edit some text which has already passed through the edit buffer and has been written to disk (to the output file). This is the purpose of backward file buffering.

Backward file buffering reads text from the output file back into the beginning of the edit buffer for further editing. First, however, it makes additional space free in the edit buffer by writing out text from the end of the edit buffer to a temporary disk file. (The file has a name extension of “.@R\$” for the main edit buffer and “.rR\$” for edit buffer ‘r’.)

Since backward disk buffering requires an additional temporary file, VEDIT requires additional free disk space when editing files.

Although backward file buffering works just as automatically and invisibly as forward file buffering, it must be used with a little more care, especially if you are using floppy disks. Since it requires an additional temporary file, you are more likely to run out of disk space.

Maximum File Size

Since editing a large file requires both an input file and an output file, the maximum file size which can be edited is 1/2 a disk.

Before editing a really large file, it may be necessary to check the file size with the QNX **ls -l** and the amount of free disk space with **df**.

When the amount of free space is *twice* the size of the file being edited, you are usually safe (unless the new file will be significantly larger than the original file). You can include any “.b” version of the file being edited in the amount of free space available. If the amount of free space is not at least equal to the size of the file being edited, you will run out of disk space even without backward file buffering.

If the amount of free disk space is barely greater than the size of the file being edited, you should avoid backward disk buffering—going to the beginning of the file when it is no longer in memory. The format of the “LINE” message on the status line tells you whether the beginning of the file is in memory.

{**GOTO, Home in Memory**} lets you go to the beginning of just the text in memory.

SUGGESTION:



If you are near the end of a very large file and need to go to the beginning, it is often faster to use {**FILE, New File**} to start editing the file over again. This has the added benefit of saving the current file.

File Locking

QNX is a multi-user multi-tasking system allowing many programs to be run simultaneously on one computer by one or more users. It is also a networked system allowing programs and users on one machine to access files on another. It therefore provides facilities to deal with the situation where one program attempts to access a file which is already in use by another program. Using these *file locking* facilities it is possible for one program to “lock out” other programs (deny them access to part or all of a file) where necessary.

VEDIT uses QNX file locking facilities. Specifically, it locks edited files against other writers while allowing other programs to read them.

This means, for example, that if you are editing a program’s source code with VEDIT, no-one else can open that file for write but anyone can read it. Another programmer trying to edit the file with VEDIT will find that his or her VEDIT reports the file as “read only” and does not allow saving of an altered version to the original filename. Commands which only read the file, such as compilers, will not be affected by the lock at all. This allows you to compile a file saved to disk while you are editing.

VEDIT file locking has the added benefit of preventing you from editing the same file in two edit buffers. (Otherwise it would be unclear which edit buffer contained the “real” file.) However, you can edit different parts of a file (using one edit buffer) in multiple windows.

When VEDIT is invoked with the **-b** option for **b**rowsing, it does not lock the files.



The POSIX standard locking facilities provided on QNX do not include *mandatory* locks which the system enforces against any and all processes, only *advisory* locks which will be respected by all programs *written* to respect them. For example, VEDIT locks prevent your files from being written by another instance of VEDIT or by any other program using locking, but do *not* guarantee protection from all programs. See the *Watcom C Library Reference* under **sopen**.

Your files are also protected by QNX access permissions (see *Utilities Reference* for **chmod**) which *do* affect all programs and by your own system administration and backup procedures.

Memory Management

This is a technical description of VEDIT memory management. It explains how memory is allocated to the edit buffers, Text Registers and Undo stacks. It explains how VEDIT operates with limited memory. Finally, it explains how the Undo capability is affected by the amount of available memory.

Memory Allocation

VEDIT can simultaneously edit several files, each of unlimited length. Although you can edit multiple large files with only about 40K of data memory per file, performance is better with more memory.

When enough memory is available, each file has its own reserved 64K segment of memory. Files up to about 60K in size are entirely in memory; larger files are partly on disk and partly in memory.

A limit on the amount of data memory VEDIT will use can be set with the `-s`*size* command-line option. (See “Starting VEDIT” in the Edit Guide - Chapter 3.) This defaults to 192K if not set.

VEDIT will continue using memory as necessary until *either* it reaches this limit *or* it asks QNX for memory and QNX refuses.

192K is the smallest value which gives you full Undo capability; at this or any larger value the Undo stacks have their own full 64K segment.

A "power user" might use a large number to support heavier use. For example, `-s512` would allow six large files to be open with about 60K of each file in memory.

Reducing the number does not eliminate any VEDIT capabilities. Even with the minimum `-s40` you can still edit several large files, use text buffers and perform undo actions. Limited memory will, however, limit the amount of text you can store in buffers and the number of undo levels available. Also, moving around within a large file or between files may entail swapping text between memory and disk. This can have a serious impact on performance.

Operating with Limited Memory

VEDIT can run with as little as 40K of data space per user. This is rarely required but may be necessary on a heavily loaded machine.

Running VEDIT in a limited amount of memory is the same as running it with the `-s` option.

VEDIT works best with lots of memory, but it can still edit a multi-megabyte file with very little memory. You may notice a considerable speed degradation.

You can also edit multiple files with very little memory. However, switching between files will involve delays for file buffering because the edit buffers will be sharing one memory segment.

Performance is best when each edit buffer has its own memory segment. Switching between edit buffers is then instant with no file buffering.

UNDO Memory Management

When edit buffers do share a memory segment, the **{FILE, Buffer Switch}** function often performs file buffering to make more memory space free for the newly active edit buffer.

The Undo facility requires memory for an Undo stack for each edit buffer and storage for the deleted text. The Undo stacks are not particularly big; each “level” requires only 4 bytes. Therefore a typical 500 level Undo stack requires 2 Kbytes of memory. However, deletions require a lot of memory. When you delete a block of 10,000 characters, 10 Kbytes of memory is needed for you to be able to undo this deletion.

When enough memory is available, the Undo facility has its own reserved 64K of memory. This memory is used for the Undo stacks for the first *two* edit buffers. The rest is used to store deleted text. Since the maximum Undo stack size is 1000 levels, up to 56K of deleted text can be stored for Undo purposes. This could be one 56K block or fifty-six 1K blocks.

Two configuration parameters determine how many undo levels are available in each edit buffer. These may be set by editing the **.cfg** files. See Chapter 7.

VEDIT uses the larger value (typically 300 to 1000 undo levels) when there is enough memory for the newly created edit buffer to have its own memory segment. It also uses this value for the first two edit buffers (including @) when there is enough memory for the Undo facility to have its own memory segment.

When a new edit buffer has to share memory with other buffers or the Text Registers, VEDIT uses the smaller value (typically 100 undo levels) in order to conserve memory space.

The Undo stacks for the first two edit buffers are placed into the Undo memory segment. This makes the maximum text space available in the first two edit buffers. The Undo stacks for additional edit buffers are placed into the same segment as the edit buffer.

The Undo facility “forgets” about deleted text when the deletion is no longer on the Undo stack, e.g. the deletion was performed more than 500 levels ago. However, the Undo facility always holds on to the last five deleted blocks on a special “deletion stack”. These blocks can be re-inserted with the {**UNDO, Delete**} function.

Since the Undo facility is restricted to available memory, it is possible, when you delete a large block of text, that there isn’t enough memory space to store the block for later Undo. When this happens you get the warning “OK TO DELETE nnnnn LINES AND RESET UNDO (Y/N)?”. If you OK the deletion, the Undo facility is also reset. This is necessary because the Undo system cannot handle the case where some editing steps are left out.

When the Undo facility has to share memory with the Text Registers or even the main edit buffer, much less memory is available for saving deleted text and you are much more likely to get the “OK TO DELETE nnnnn...” message.

Keyboard Layouts

VEDIT Default Keyboard Layout

The entire default VEDIT keyboard layout is listed below in alphabetic order by key name. It includes the actual contents of each Keystroke Macro. This list is similar to that printed by {**MISC, Keyboard Layout, Print**}.

<Alt>-	[MENU] B
<Alt>-<C>	[MENU] C
<Alt>-<E>	[MENU] E
<Alt>-<F>	[MENU] F
<Alt>-<G>	[MENU] G
<Alt>-<H>	[MENU] H
<Alt>-<I>	[CANCEL][VISUAL EXIT]
<Alt>-<K>	[MENU] MKR

<Alt>-<L>	[MENU] MV
<Alt>-<M>	[MENU] M
<Alt>-<N>	[MENU] FN
<Alt>-<O>	[MENU] FON
<Alt>-<P>	[MENU] P
<Alt>-<R>	[MENU] MC
<Alt>-<S>	[MENU] S
<Alt>-<T>	[CANCEL][VISUAL EXIT]
<Alt>-<U>	[MENU] U
<Alt>-<V>	[MENU] WV
<Alt>-<W>	[MENU] W
<Alt>-<X>	[MENU] FE
<Alt>-<Y>	[MENU] FOYH
<Alt>-<F1>	[HELP]
<Alt>-<F2>	[REPLACE]
<Alt>-<F3>	[MENU] GM
<Alt>-<F4>	[MENU] WS
<Alt>-<F7>	[MENU] UE
<Alt>-<F5>	[DEL PREV WORD]
<Alt>-<F6>	[DEL NEXT WORD]
<Alt>-<F8>	[VISUAL ESCAPE]
<Alt>-<F9>	[BLOCK MOVE]
<Alt>-<F10>	[T-REG MOVE]
<Backspace>	[BACKSPACE]

<Ctrl>-	[SCROLL UP]
<Ctrl>-<C>	[CANCEL]
<Ctrl>-<E>	[VISUAL EXIT]
<Ctrl>-<F>	[MENU] EF
<Ctrl>-<G>	[MENU] GG
<Ctrl>-<K>	[MENU] MKA
<Ctrl>-<N>	[MENU] BN
<Ctrl>-<P>	[MENU] PB
<Ctrl>-<Q>	[ENTER CTRL]
<Ctrl>-<R>	[REPEAT]
<Ctrl>-<S>	[MENU] GS
<Ctrl>-<U>	[MENU] UL
<Ctrl>-<V>	[SCROLL DOWN]
<Ctrl>-<X>	[ERASE LINE]
<Ctrl>-<Z>	[ERASE EOL]
<Ctrl>-<Backspace>	[VISUAL EXIT] OK
<Ctrl>-<↓>	[SCROLL DOWN]
<Ctrl>-<←>	[SCROLL LEFT]
<Ctrl>-<→>	[SCROLL RIGHT]
<Ctrl>-<↑>	[SCROLL UP]
<Ctrl>-<End>	[SCREEN END]
<Ctrl>-<Enter>	[NEXT LINE]
<Ctrl>-<F5>	[PREV PARAGRAPH]
<Ctrl>-<F6>	[NEXT PARAGRAPH]

<Ctrl>-<F8>	[MENU] ME
<Ctrl>-<F9>	[BLOCK COPY]
<Ctrl>-<F10>	[T-REG COPY]
<Ctrl>-<Home>	[SCREEN BEGIN]
<Ctrl>-<PgDn>	[MENU] GE
<Ctrl>-<PgUp>	[MENU] GB
<↓>	[CURSOR DOWN]
<←>	[CURSOR LEFT]
<→>	[CURSOR RIGHT]
<↑>	[CURSOR UP]
	[DELETE]
<End>	[LINE END]
<Enter>	[RETURN]
<Esc>	[ESCAPE]
<F1>	[MENU]
<F2>	[FIND]
<F3>	[FIND AGAIN]
<F4>	[MENU] FB
<F5>	[PREV WORD]
<F6>	[NEXT WORD]
<F7>	[UNDENT]
<F8>	[INDENT]
<F9>	[MENU] BS
<F10>	[T-REG INSERT]

<gray />	/
<gray *>	*
<gray ->	-
<gray +>	+
<Home>	[LINE BEGIN]
<Ins>	[INSERT TOGGLE]
<PgUp>	[PAGE UP]
<PgDn>	[PAGE DOWN]
<tab>	[TAB CHARACTER]

Keystroke Macro Examples

This appendix lists a few Keystroke Macros that you can add to VEDIT. These and many more are listed in the supplied library files **keymac.lib** and **macros.lib**. The VEDIT macro language is used in these Keystroke Macros.



Although many Keystroke Macros examples in this manual are listed on several lines, they should always be entered as one long line. In particular, don't press <Enter> after you press the key assigned to **VISUAL EXIT** (typically <CTRL-E>).

Color Selection

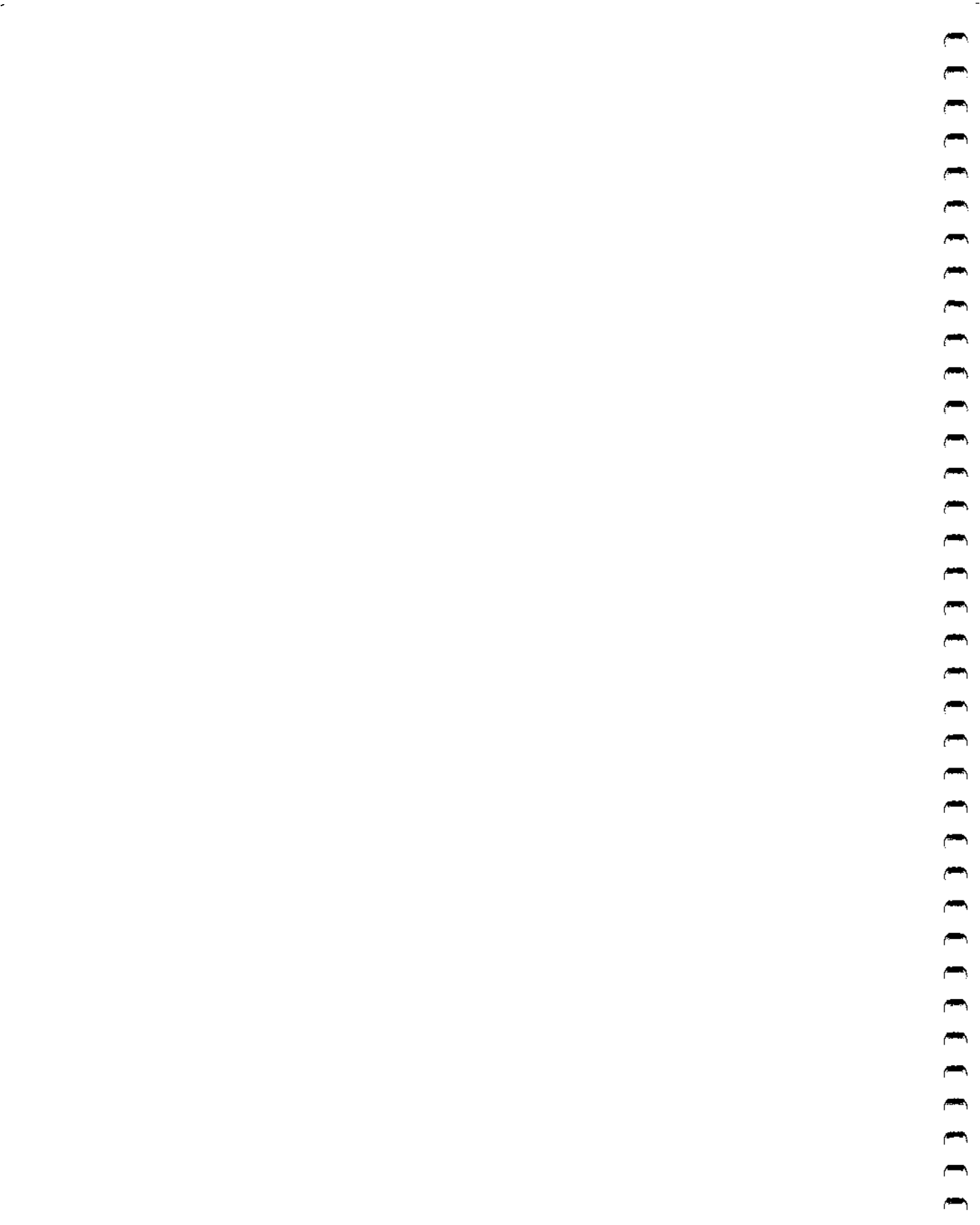
Within VEDIT you can toggle between about 15 built-in color combinations with **{MISC, Color Toggle}**. Alternatively, the following Keystroke Macro displays a chart of all colors and lets you enter the value of the desired color.

```
[VISUAL EXIT]
.wzXS(100) Q100=0[YWZ] .waXS(101) 0XS(102) YEC
YEN 128[Q102YEA :XT(102) XA(102)] Q101YEA 2YEN
XQ(101)"Enter value of desired color: "
Q101YEA Q100=0[YV"\ME\WZ"] -V
```

Word Count

The following Keystroke Macro counts the number of words in a file and displays the result. You can assign it to any desired key.

```
[VISUAL EXIT]
.pXS(100) +YT"Counting words in file. Please
wait..." _B #:$N"|s|a" .rvXS(101) Q100EJ
+YWS(?) YEC YT"Word count = " :XT(101)
XK(100)"      Press any key..."
```



Index

!

"1-END" message 36
 <CR> 49, 206
 <LF> 206
 ".@R\$" file 247
 [and] 71
 \ (regular expression escape character) 72
 | 72, 76, 182
 IR 82
 { and } 72

A

Abandoning file (See also File - Quit) 32
 ASCII Table 157
 Assembly language programming 51, 80
 Auto-buffering 246
 Auto-execution (of macros) 106
 Auto-indent mode 84, 202
 Auto-startup 104

B

[BACKSPACE] 213
 Backup file ("b") 33, 206, 246
 Backward file buffering 247
 Beginning of file/buffer 145, 146
 Block 58, 59, 60, 61, 62, 63, 64, 65, 66, 67
 Characters included 60
 Columnar blocks 58, 63
 Copy/move (See Copying/moving text) 58, 59,
 60, 61, 62, 63, 64, 65, 66, 67
 Deleting 124
 Indenting 58, 216
 Searching 185
 Write to disk 125
 { BLOCK } menu 116, 117, 118, 119, 120, 121,
 122, 123, 124, 125, 126
 [BLOCK COPY] 61, 119, 213
 Block markers 150
 Auto-cancel 204
 Change size of block 126
 Remove 125, 204, 211
 Setting 59, 117
 Setting columnar markers 118
 "BLOCK" message 34
 [BLOCK MOVE] 61, 120, 213
 Browse Mode 30, 135
 Buffer (See Edit buffer) 138

C

- [CANCEL] 38, 214
- “Caps” lock 35
- Carriage-return character 49, 206
- Center line 130
- Change directory 140
- “COL” message 35
- Color
 - Chart 225, 256
 - Configuration 224
 - Edited text 160, 256
 - Toggle 160
- Column
 - Display line # 34
 - Go to column # 149
- Columnar block (See Block) 58, 118
- Command Macros 99, 100, 101, 103, 105, 107, 109, 111, 113
 - Auto-execution 30, 104, 106
 - Execute 101, 161, 163
 - Load 101, 161
 - Text Register usage 102
- Command Mode 100
 - Enter via Keystroke Macro 103
 - Message on status line 35
- compare.vdm file 110, 111, 112
- Conditional conversion character 51, 202
- Configuration 221, 223, 225, 227, 229, 231
 - Overview 199
 - Read from Disk 210
 - Save to Disk 208
 - {CONFIG} menu 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209
- Continuation character 57
 - Configuration 228
- Continuation line 57
- Control characters
 - Display 48, 207
 - Entering 48, 128
 - Keyboard 47
 - Search for 186
- Convert lower to upper case
 - (See Lower and upper case) 51
- Copying/moving text 60, 192, 213
 - One file to another 93
 - Text registers 61
 - To a Text Register 121, 122
 - To the cursor 119, 120
- Current directory 140, 154
- Cursor
 - Blink rate 207
 - Display (style and type) 53, 206
 - Position after block insertion 205
 - Positioning mode 203
- [CURSOR DOWN] 214
- [CURSOR LEFT] 214
- Cursor movement
 - By paragraphs 17, 217
 - By screens/pages 17, 217
 - By sentence 103
 - By words 16, 217
 - In dialog boxes 21, 40
 - In menus 38
 - In Point and Shoot 41
 - In text 13, 16, 17, 214, 227
- [CURSOR RIGHT] 214
- [CURSOR UP] 214
- Cut and Paste (See Copying/moving text) 60

D

- Date display 154
- Default directory 140, 154
- Delete
 - Block of text 124
 - Large blocks 191
 - Line 129, 130
 - Text (Overview) 13, 14

- Undo 191, 192
- Windows 96
- Words 215
- [DEL NEXT WORD] 215
- [DEL PREV WORD] 215
- [DELETE] 215
- Dialog boxes 39, 40
 - Line editing/recall 21, 39
- Directory
 - Change default 140
 - Default 154
 - Display 20, 140
 - Selection 42
- Disk full error (recovery) 239
- "DISK" message 35
- Disk space
 - Maximum file size 247
 - Usage 247
- DOS text file 50, 206
- Duplicate line 103

E

- Edit buffer 89
 - Active (current) 34
 - Multiple file editing 137
 - Name 62
 - Name on status line 139
 - Switching 138, 251
 - Text Register differences 93
 - Usage 155
 - Window usage 96
- Edit Function
 - Keyboard layout 168
 - Reference 213, 215, 217, 219
- {EDIT} menu 127, 128, 129, 130, 131, 132, 133
- Editing
 - Multiple files 88, 89, 90, 91, 92, 93, 136, 249
 - New file 135
 - One file in two windows 97
 - Prompts (See Dialog boxes) 39
 - Switching between files 91, 138
 - With windows 92
- Emulation
 - Modes 52
 - Of other editors 202
- End of file/buffer 147, 148
- End of line character (See Newline character) 49
- [ENTER CTRL] 128, 215
- <Enter> key 12, 204, 206
- Entering new text 12
- [ERASE LINE] 129
- Erase (See Delete) 129
- [ERASE EOL] 130, 215
- [ERASE LINE] 215
- Error messages 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243
- <Esc> key 38, 171
 - Assignment to [ESCAPE] 172, 211
 - Enter into text 128, 167
 - Escape sequences 172
 - Usage on terminals 172
- [ESCAPE] 215
- {ESCAPE} menu 26, 210, 211
- Escape sequence 171
- Exit
 - VEDIT 27, 32, 33, 91, 143, 211

F

- File
 - Altered/unaltered 33, 134, 136, 143, 191
 - Buffering 246
 - Change name (Save as) 141
 - Comparison (macro) 110, 111, 112
 - Handling/management 245, 246, 247, 248
 - Input file 154, 245
 - Inserting 140

- Large (long) 246
- Maximum file size 247
- Names of files being edited 35, 155
- Output file 154, 245
- Quit (abandon) and exit 32, 143, 211
- Read-only 134, 236
- Save and continue editing 26, 142
- Save and exit 27, 32, 143, 211
- Selection (See Point and Shoot) 41
- {FILE} menu 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144
- File locking 248
- Filename (displaying) 35
- Filespec (Point and Shoot) 41, 42
- [FIND] 215
- [FIND AGAIN] 216
- Form-feed character 179, 181
- Format paragraph 86, 131
- Free memory space 154
- Full disk (See Disk full error) 239
- Function/control keys 47, 169, 171, 172, 207

G

- {GOTO} menu 145, 146, 147, 148, 149, 150, 151
- Graphics characters
 - Display 48, 207
 - Entering 48
 - Keyboard 53
 - Search for 186

H

- Help 42, 216
 - Help level 208
 - Introduction 21, 22, 23, 24, 153
 - Keyboard layout 22, 43, 153
 - Keystroke Macros 43

- On-line 21, 22, 23, 24, 42, 43, 153, 216
- Status display 153
- Topics 24, 43, 153
- {HELP} menu 152, 153, 154, 155
- High bit characters (See Graphics characters) 48
- Highlighting (See Block markers) 59
- Horizontal scroll increment 205
- Horizontal scroll margin 57, 205
- Horizontal scrolling 56, 205
- Hot-keys (See Keystroke Macros) 37

I

- [INDENT] 59, 84, 216
- Indent increment 84, 202
- Indenting text 84, 219
- Input file 154, 245
- “INS” message 35
- Insert
 - File 140
 - From ASCII Table 157
 - Text (Overview) 12
- Insert mode 12, 40, 127, 204
- [INSERT TOGGLE] 127, 216
- Invoking VEDIT 29, 30, 31, 104
 - Options 30, 106
 - Overview 12

J

- Justifying paragraphs 86, 87, 201

K

- Keyboard
 - Notation 8, 9
 - Options 53, 207

Keyboard layout
 CRT terminals 171
 Keyboard layout 163
 Change 168, 173, 176
 Display/help 43, 153, 172
 Edit (Keyboard layout file) 176
 Edit Functions 168
 Load from disk 104, 174
 Print 173
 Save to disk 175
 UNIX 171
 Default layout 252
 Keystroke macros 53, 54, 55
 Add/Record 55, 102, 165, 167
 Built-in 43
 Display 172
 Escape sequence 164
 Examples 55, 256
 Help 43, 153
 Hot-keys 54
 Macro language 54, 102
 Misc. notes 164, 256
 Modify 166
 Print 173
 Repeating 167
 Save 208

L

Large files (See File - Large) 246
 Left margin 84, 201
 Line
 Erase 129
 Center 130
 Display # 34
 Editing/recall 40
 Erase 130
 Go to line # 148
 Wrap long lines on screen 57, 205

[LINE BEGIN] 16, 216
 [LINE END] 16, 217
 "LINE" message 35
 Line-feed character 206
 See also Newline character 49
 Lower and upper case
 Converting keys 51, 202
 Converting text 133
 Reverse case (Keyboard) 53, 207
 Searching 186, 189
 Switch case of text 134

M

Macro language (See Command Macros) 99
 Margin (See Left and Right margins 84
 Markers
 (See Block markers or Text Markers) 149
 Match parentheses 151
 Maximum file size 247
 Memory
 Management 249, 250, 251
 Restricting usage 30
 Space free 154
 [MENU] 217
 Menu system
 Using 19, 37, 38
 {MISC} menu 156, 157, 158, 159, 160, 161,
 162, 163, 164, 165, 166, 167, 168, 169, 170,
 171, 172, 173, 174, 175
 "MORE" message 36
 Mouse Support 44, 45, 46
 Block operations 46
 Cursor options 208
 Menu access 44
 Notes 47
 On-line help 44
 Quick cursor movement 45
 Relative jump in file 46

Search/replace options 47
Multiple drives 247
Multiple file processing 108, 109
Multiple files (See Editing) 88, 89, 90, 91, 92, 93

N

Network
 Printing 178
 “NEW FILE” message 29
Newline character 50, 206
 Display 49, 208
 Searching 77, 186
[NEXT LINE] 217
[NEXT PARAGRAPH] 17, 217
[NEXT TAB STOP] 217
[NEXT WORD] 16, 217
Notation 8, 9
Null character 186
“Num” lock 35

O

Offset paragraph 87
On-line help (See also Help) 42, 43
Operator precedence 75
Output file 154, 245
Overstrike mode 127, 204

P

[PAGE DOWN] 17, 217
[PAGE UP] 17, 217
Paragraph
 Definition of 83
 Formatting 86, 131, 201
 Justifying 86, 87

Offset 87
Unjustify 88
Parentheses matching 151
Pattern matching 76
 Codes 76
 Pattern sets 81
Point and Shoot file selection 41
 Change directory 140
[PREV PARAGRAPH] 217
[PREV WORD] 217
[PREVIOUS PARAGRAPH] 17
[PREVIOUS WORD] 16
Print
 Block 177
 Cancel/abort 177
 Configuration 180
 Eject Page 178
 Entire file 107, 177
 Flush print job (network) 178
 Form-feed character 179, 181
 Formatter (macro) 107
 Introduction 25, 176
 Keyboard layout 173
 Laser Printer notes 180
 Line spacing (double, etc.) 181
 Macro 107
 Margins 180, 181
 Page perforations 179
 Paper length 180
 Printer port selection 181
 Problems 179
 Reset Printer 179
 {PRINT} menu 176, 177, 178, 179, 180
print.vdm file 106, 107
Prompts (See Dialog boxes) 39, 40

Q

QNX text file 206
 Quit (abandon) and exit 211
 Quit (abandon) and exit 32, 143

R

Read-only file 134
 Regular expressions 68, 189
 [REPEAT] 132, 217
 Keystroke Macros 167
 Repeating operations (See also [REPEAT]) 132
 [REPLACE] 218
 Replacement string 187
 [RETURN] (See also <Enter> key) 218
 Right margin 85, 201
 "RM" message 36
 Running QNX programs 158, 159

S

Save text and exit 27, 32, 143, 211
 Screen
 Color/attributes 30, 224
 Display (updating) 53
 Initialize 199
 Overlap when paging 227
 Scrolling 56, 57
 Size 226
 [SCREEN BEGIN] 218
 [SCREEN END] 218
 [SCROLL DOWN] 56, 218
 Scroll increment 205
 [SCROLL LEFT] 57, 205, 218
 "Scroll" lock 35
 Scroll margin 205
 [SCROLL RIGHT] 57, 205, 218

[SCROLL UP] 56, 218
 Scrolling 15, 56, 57, 218, 227
 Search 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 181, 182, 183, 184, 185, 186, 187, 188, 189, 215
 Again 183, 188
 Case sensitive 186, 189
 Configuration 189
 Far (to end of large file) 183
 For words 184
 From beginning of file 183
 In blocks 185
 Introduction 182
 Near (only to end of memory) 184
 Options 183
 Regular expressions 184, 189
 Reverse (backwards) 184
 String 82, 182, 215
 Text 182
 {SEARCH} menu 181, 182, 183, 184, 185, 186, 187, 188, 189
 Search and Replace 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 181, 182, 183, 184, 185, 186, 187, 188, 189, 218
 Again 188
 Options 188
 Security 30, 104, 106
 Set markers
 (See Block markers or Text markers) 117
 Shell 158, 159
 Shelling out 158
 SORT.vdm file 113, 114
 Sorting (macro) 113, 114
 Spaces
 Trimming 205
 Convert to Tabs 64, 67
 Trailing 64
 Trimming 66
 Starting VEDIT 29, 30, 31
 Starting VEDIT (See Invoking VEDIT) 12

Status display 153
Status line 34, 35, 36
 Configuration 228
Strip comments 80

T

[T-REG COPY] 121
[T-REG INSERT] 123, 219
[T-REG MOVE] 122, 219
[TAB CHARACTER] 204, 219
Tab character (key) 50, 200, 219
 Convert spaces to tabs 64, 67
 Convert Tab character to spaces 64, 67
 Display 49, 50, 208
 Expand Tab key to spaces 51
Tab stops 50, 200
Temporary disk file (“.@R\$”) 247
Text Markers
 Go to 150
 Setting 149
“TEXT” message 36
Text Register
 Usage by Command Macros 102
 As an Edit buffer 89, 93
 Copy/move to 61, 121, 122
 Display 156
 In search string 82
 Inserting 62, 123
 Memory usage 154, 156
 Naming 62
 Overview 61
 Usage 62, 155
Time display 154
Trailing spaces 64
Typewriter emulation mode 204

U

[UNIDENT] 59, 84, 219
Undo
 Deletion 124, 191, 192
 Edit 191
 Introduction 18, 190
 Levels 190, 251
 Line 191
 Memory management/limitation 251
 Reset 193, 252
 {UNDO} menu 190, 191, 192
UNIX text file 206
Unjustify (See also Justifying paragraphs) 88
Upper and lower case
 (See Lower and upper case) 51
Using Menus (See Menu system, using) 19

V

.vdm file 101
veditrc file 104, 106
Version number (VEDIT) 154
Vertical scrolling 56
vhelp.hlp file 43
[VISUAL ESCAPE] 220
[VISUAL EXIT] 103, 220
Visual Mode 220
 Definition 99

W

wildfile.vdm file 108, 109
Windows 94, 95, 96, 97
 Name 155
 And edit buffers 96, 198
 Attach to edit buffer 96
 Border characters 226

- Create 95, 194
- Deleting 96, 196
- Editing multiple files 92, 137
- Multiple per file 197
- Name 195, 198
- Overview 94
- Remove all 199
- Splitting 197
- Switch to another window 197
- Zooming 97, 199
- { WINDOW } menu 193, 194, 195, 196, 197, 198
- Word (Definition of) 83
- Word count 257
- Word Processing 82, 83, 84, 85, 86, 87
 - Configuration 201
- Word wrap 85

