VEDIT 5.21

Fully Configurable
Multiple File Text Editor
Universal File Editor

User's Manual

Greenview Data

VEDIT

Universal File Editor
For Text, Program, Database, Binary
And Mainframe File Editing
Version 5.21

Manual Written By:
Theodore Green & Charles Scott

Programmed By:
Theodore Green & Thomas Burt

Greenview Data, Inc.
2773 Holyoke Lane
Ann Arbor, MI 48103
Telephone: (734) 996-1300
Sales: (800) 458-3348
Fax: (734) 996-1308
E-Mail: support@vedit.com
Website: www.vedit.com

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.

Last Full Manual Revision: June 22, 1999
(Minor Revision: Jan. 17, 2002)

ACKNOWLEDGEMENTS

We would like to thank the following people for their assistance.

Christian Ziemski for the exceptionally thorough beta-testing of new versions and assistance in setting up the Discussion Conference on our Web site.

Scott Lambert for the numerous suggestions, for supporting other users in the Discussion Conference, and for running his own "VEDIT macro" Web site.

Maxim Glukhov for writing the new BOX-DRAW.VDM and ASCII2.VDM macros supplied with VEDIT.

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

Peter Freed of Data Base Management Systems, Inc. for writing the CMD-CONV.VDM, DBASE.VDM and WS6.VDM (enhanced WordStar emulation) macros supplied with VEDIT.

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

TRADEMARKS

TABLE OF CONTENTS

Chapter 1 - Introduction..... 9
Welcome to VEDIT.....9
Main Features.....11
Ready-To-Use Macros.....13
V-SPELL.....16
System Requirements.....17
Using this Manual.....18
Notation.....19
Product Support.....20

Chapter 2 - Getting Started..... 21
Installation.....21
Windows Installation.....22
Un-installing VEDIT.....24
DOS Installation.....25
DOS Automated Installation.....25
DOS Manual Installation.....25
Hardware Configuration (DOS Version).....26
Installing DOS VEDIT in Windows.....27
Installing DOS VEDIT in IBM OS/2.....28
Testing the VEDIT DOS Installation.....29
CONFIG.SYS and AUTOEXEC.BAT Files.....30
Network Installation.....31
Initial Configuration.....33
Description of Files.....34
Keyboard Layout.....38

Chapter 3 - Quick Tutorial..... 43
Starting VEDIT.....44
Entering New Text.....45
Deleting Text.....46
Moving the Cursor.....47
Undo and Redo.....49
Repeating Operations.....50
“Cut and Paste” a Block.....50
Printing Text.....53
Saving Your Work.....54
Exiting VEDIT.....54
That’s It!.....54

Chapter 4 - Editing Guide 55

Starting (Invoking) VEDIT 55
Starting VEDIT for Windows 55
Starting VEDIT for DOS 57
Invocation Options (All Versions) 57
"VEDIT" Environment Variable 60
Loading Multiple Files 60
Read-only (Browse-only) Mode 61
Overwrite-Only Mode 62
Exiting VEDIT 63
Edit Session Restore 64
Backup Files 65
Auto-file Save 66
The Status Line 67
User Interface 69
Selecting Display Fonts (Windows version) 69
Shortcuts and Suggestions 69
DOS, UNIX and QNX Versions 70
Scrolling the Screen 71
Vertical Scrolling 71
Horizontal Scrolling 71
Wrapping Long Lines on the Screen 72
Screen Display & Keyboard Characters 73
Entering Control and Graphics Characters 73
Control and Graphics Character Display 74
Display Modes 74
The <Tab> Key and Tab Characters 75
Lower and Upper Case Conversion 76
Key Emulation Modes 77
Other Keyboard Input Options (DOS Only - Technical) 78
Other Screen Display Issues (DOS Only - Technical) 78
Hex Mode Editing (and Octal) 79
Searching in Hex, Decimal or Octal 80
Entering Numbers in Hexadecimal 80
Undo and Redo 81
File Types - Win/DOS, UNIX, Mac, Binary 82
Windows/DOS and UNIX Text Files 83
Macintosh Text Files 83
Binary/Data Files (Record Mode) 84
Database Files With Headers 85
Editing Very Long Lines (Technical) 85
Keystroke Macros 86
Recording Keystroke Macros 87
Adding Keystroke Macros 88
Adding Keystroke Macros from KEY-MAC.LIB 89

Modifying an Existing Keystroke Macro....90
Deleting Keystroke Macros....90
Editing the Keyboard Layout....91
Adding a Keystroke Macro from KEY-MAC.LIB....92
Loading a New Keyboard Layout....93
Block Operations....94
Marking (selecting) a Block of Text....94
VEDIT's Blocks are Persistent....96
What exactly does the block include?....96
{BLOCK, Copy / Move to cursor}....97
Text Registers and the "Scratchpad"....98
The "Scratchpad" Text Register....98
Accessing Other Text Registers....99
Block options - Fill and Overstrike....99
Emptying a Text Register....100
Text Register Usage....100
Cut & Paste Huge Blocks....103
The Windows Clipboard....103
Block Indenting....104
Columnar Blocks....105
Translating a Block or File....110
Translating between EBCDIC and ASCII....110
Translating between ANSI and ASCII....111
Creating Your Own Translation Table....111
Sorting Lines in a Block / File....112
Sorting by multiple fields (major and minor keys)....112
Printing in VEDIT....113
Basic Operation....114
Printer Margins....115
Using the PRINT.VDM Macro....115
Print Display Mode....116
EBCDIC and other Translate Tables....116
Print "Jobs" and [Finish/Eject]....117
Print Job Start/Finish Strings....118
Search and Replace....120
Pattern Matching....120
Matching the "Newline" - with "|L" and "|N"....122
Matching the Beginning/End of a Line with "|<" and "|>"....123
Matching Multiple Characters with "|M" and "|Y"....123
Pattern Sets....124
Using Text Registers in Search Strings....125
Regular Expressions....126
Regular Expression Basics....126
Special Matching Characters....129
The "OR" Operator....129

Groups and Replacement Strings 130
Complete Examples 131
Matching the “Newline” 131
Maximize Regular Expression Matching 132
Word Processing Functions 133
Definition of “Word” and “Paragraph” 133
Indenting Text (Left Margin) 134
Word Wrap (Right Margin) 135
Formatting and Justifying Paragraphs 136
Offset Paragraphs 136
Justification 137
Format Paragraph Options 137
Editing Multiple Files 138
{FILE, Open (More)} Sub-menu 139
Switching Between Files (Edit Buffers) 140
Closing Files and Windows 140
Copying Text From One File to Another 141
Starting (Default) Directory for File-Open 142
Edit Buffer Details 143
Windows 144
Introduction 144
Windows are attached to buffers 144
How window names are displayed 144
Switching Between Windows 145
Zooming A Window 145
“Full-Size” Windows (Windows version) 146
Editing One File in Two Windows 146

Chapter 5 - Advanced Topics 147

STARTUP.VDM File 147
Changing Configuration with STARTUP.VDM 149
Using a different startup file 149
Name of STARTUP.VDM and VEDIT.INI 150
{USER} and {TOOL} Menus 151
File-type Configuration 153
Color Syntax Highlighting 155
Automatic syntax highlighting for C 155
Automatic syntax highlighting for HTML 156
Automatic syntax highlighting for other languages 157
Creating your own “.SYN” syntax definition file 157
Template Editing 158
Manual Setup 158
Automatic template editing for C and HTML 158
HTML Editing Features 160

Command Macros 161
VEDIT PLUS Macro Language 161
Command Macros and Text Registers 162
Loading and Executing Command Macros 163
Auto-Execution 164
PRINT - Print Macro 165
WILDFILE - Multi-file Processing 166
Auto-executing the WILDFILE.VDM macro 167
COMPARE - Compare Files 168
COMPDIR - Compare Directories 170
SORT - Sorting Macro 171
Running SORT.VDM via Windows "Run" command 172
DBASE.VDM Macro 173
Optional "Hot-key" for xBase Files 174
CFUNC - C Program Outliner 175
RUNSHELL - Run other programs 176
"ctags" Symbol Lookup
Setup 178
Usage 179
Advanced Usage Notes 179
Integrated Compiler Support 180
Overview 180
Compiler Support Installation 181
Enable Compiler Support 181
Configuring COMPILE.CNF (or JAVA-SDK.CNF) 182
Running the Compiler Support 183

Chapter 6 - Menu Reference 185

File Menu 186
Edit Menu 196
Undo (Sub-menu) 203
Scratchpad (Sub-menu) 206
Delete (Sub-menu) 208
View Menu 211
Block Menu 218
Edit/Translate (Sub-menu) 227
Convert Newlines (Sub-menu) 233
Goto Menu 235
Misc Menu 241
Search Menu 249
Window Menu 258
Config Menu 263
Keyboard Layout (Sub-menu) 296
Display unused keys 299
Help Menu 302

Escape Menu 307
Mouse Right-Click Menu 310

Chapter 7 - Edit Function Reference 311

Chapter 8 - Configuration 317
Basic Configuration 318
How VEDIT Configures Itself..... 318
Troubleshooting 320
VEDIT.KEY Layout File 322
Modifying the VEDIT.KEY file 323
Configuration Commands in ".KEY" Files 323
VEDIT.CFG Configuration File..... 324
List of Config( ) Parameters 325
Config_String( ) Parameters 337
Config_Tab( ) Parameter..... 338

Chapter 9 - Messages.....339

Appendices - 351
A - File Management 351
Basic File Handling..... 351
Automatic File Buffering 352
Maximum File Size 352
Networking and Multi-Tasking..... 353
B - Search Modes Summary 354
Pattern Matching Codes 354
Regular Expressions..... 355
C - Application Notes 357
D - Troubleshooting (DOS)..... 361
E - IBM PC Keyboard Layout 363
F - IBM PC Color Chart 367
G - ASCII Table..... 368

Chapter 1
Introduction

Welcome to VEDIT

Purpose of Program

VEDIT is an editor designed not only for text preparation and program development, but also for editing large database, mainframe and binary files. It can edit in ASCII, EBCDIC, Hexadecimal and Octal, and supports variable length and fixed length database records.

VEDIT lets you perform near-miracles on data files. If you have ever had to patch a corrupted database file, convert a huge mainframe file to a DOS/Windows text file, translate between ASCII, EBCDIC, ANSI and custom character sets, examine a Postscript file, perform a last minute search/replace on a 400 Megabyte file, or search/replace a thousand files, you probably wished for the kind of speed and capability that only VEDIT offers.

As a text editor, VEDIT is intended for both program development and text preparation. It is ideal for writing programs (e.g. C, Pascal, Basic, Assembler), HTML and lengthy documents such as reports or manuscripts. It is also well suited for the preparation of text files being used with Desktop Publishing packages such as Corel's Ventura Publisher (tm).

Since VEDIT can efficiently edit *any* file you will ever encounter, including binary/data files and huge files up to 2 Gigabytes (2000 Megabytes) in size, it is ideal for editing and translating files downloaded from Mainframe computers and CD-ROM data files. It effortlessly handles database (e.g. xBASE .DBF), postscript, plotter output, and other non-standard files. And VEDIT is the fastest editor available for huge multi-megabyte files.

VEDIT can process entire groups of files automatically, even thousands of files. The same edit changes, e.g. a search and replace, can be applied to all files in a directory, or even in all subdirectories.

Advanced features include multi-file, multi-window editing, template editing, syntax highlighting, search/replace with pattern matching or regular expressions, and a full range of block operations by character, line, or column. The entire keyboard layout and over 200 options are fully configurable.

Programming features include parentheses matching, smart indenting, and language specific color syntax highlighting and template editing for C, Basic, Assembly language, HTML and others. Additional features implemented as

macros include “Ctags” lookup, C/Pascal program outlining, and integrated support for numerous compilers, linkers, debuggers and Make.

Simple, Yet Powerful

VEDIT is simple enough to learn and use for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. It offers a rich assortment of editing capabilities, simple menu operation, complete on-line help facilities, and the ability to edit text and binary files up to 2 Gigabytes in size.

VEDIT’s multi-mode editing and display capabilities let you effortlessly edit any file in the most efficient manner possible. Edit in ASCII, EBCDIC (used by IBM mainframes), Hexadecimal or Octal modes, or split the screen for any combination of modes. File modes support Windows/DOS text, UNIX text, Macintosh text, binary and many fixed-length record data formats.

A block or entire file can be translated between ASCII and EBCDIC, ASCII and ANSI graphics characters, or with a custom translation table. Text files are easily converted between Windows/DOS, UNIX and Macintosh “newlines” (Carriage- Return and/or Line-Feed). Fixed-length records can be converted into normal text files and vice versa.

A file/block can be sorted using any desired columns (fields) as the sort key(s). A Sort macro is supplied for sorting multi-line mailing lists.

Totally Configurable, Yet Instant Startup

The comprehensive configuration lets you completely determine your own keyboard layout and precisely configure VEDIT to your particular hardware, applications and personal preferences. Over 200 configuration options are available. By using the available options and “*keystroke macros*”, VEDIT can very closely emulate other editors and word processors.

VEDIT starts up instantly; with its speed, almost every operation is performed instantly. Written mostly in 32-bit assembly language, VEDIT is exceptionally small and uses no overlays or .DLLs. The Windows version 32-bit VPW.EXE is only 420K; the DOS version VEDIT.EXE is only 135K.

VEDIT Family of Editors

VEDIT is one of the VEDIT Family of Editors which also includes VEDIT PLUS and VEDIT Jr. The VEDIT family is available in both Microsoft Windows and DOS versions. VEDIT PLUS is also available for SCO UNIX and QNX.

Attractive pricing is available for additional site licensing and networks. A cost effective software subscription program is available.

Main Features

- Multiple file editing. Simultaneously edit up to 32 files, each up to 2 Gigabytes (2000 Megabytes) in size. Efficiently edits huge text and binary/data files.
- Multi-mode editing. File modes support DOS, UNIX and Mac text, IBM's EBCDIC, binary and many fixed-length-record formats.
- Flexible windowing. (Windows version uses "MDI" type windows; DOS version simulates it.) Any file can be viewed in any window, or different parts of one file can be displayed in separate windows. Windows can be tiled, cascaded or zoomed.
- Multi-mode display. Each window can display in five different ASCII modes, hexadecimal, octal or EBCDIC. Or split the screen to view any file in multiple modes at once.
- File translating. Each character in a block or an entire file can be translated according to a custom translate table. Tables for translating between ASCII and EBCDIC, and between IBM PC and ANSI (Windows) graphics characters are supplied.
- File conversion. All newlines (Carriage-Return and/or Line-Feed) in a block or file can be converted between the type use for DOS/Win, UNIX and Macintosh text files. Fixed-length record files can be converted into text files and vice versa.
- Stream, column and line blocks. VEDIT has every type of block operation imaginable. Blocks may be moved or copied directly, or 100 scratch-pad buffers can be used for extensive "cut and paste" operations. Blocks can be copied from one file to another. Clipboard is used to exchange blocks with other Windows programs.
- More block functions. Switch blocks to upper/lower case, fill blocks, insert empty blocks, strip high bits and much more.
- Powerful search and replace. Supports sophisticated pattern matching, regular expressions, reverse searching, selective and global replace. Also incremental searching, search all open files, open all files containing a search string, or compare two files.
- WILDFILE macro permits search/replace in large groups of files in a directory or all subdirectories. Can even perform numerous search/replace on all files.
- Undo and Redo. Up to 1000 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.
- Auto-save of files. Automatically saves all modified files after a configurable number of minutes. Helps prevent losing your work due to power failures, etc.

- Automatic indenting simplifies editing of structured programs; advantageous for word processing. Block indent/undent permits quick restructuring. Parentheses matching aids C programming.
- Execute single DOS commands (DIR, compilers, V-SPELL, etc.) or enter DOS and return later.
- Flexible printing with variable margins and page size. Print a block of text or the entire file with variable spacing. Windows version supports a selectable printer font and size. PRINT macro adds a convenient header (file name, date and page number).
- ASCII table. Pop up the ASCII table or insert any desired control, printer or graphic character into the text.
- Horizontal scrolling for editing long lines. Optionally, long lines can be wrapped onto multiple screen lines.
- Word processing. Word wrap, paragraph formatting between adjustable margins, optional justification, printing. (Does not enter any special control characters into the text.) Can read/write WordStar compatible files.
- Integrated compiler support. Popular compilers, assemblers, linkers, debuggers and Make programs can be run from within VEDIT. The cursor will be automatically positioned on source code lines containing compilation errors.
- Color syntax highlighting displays different logical parts of a program in different colors. For C, HTML, Pascal, Basic, Perl, Clipper, Systat and others. Users can set up syntax highlighting for other languages or applications.
- Template editing performs shorthand expansion by recognizing a keyword and expanding it to the full string of characters. For C and HTML.
- The “Normal” keyboard is very similar to Win95 Microsoft products. Or chose a very compatible WordStar, Word Perfect or Brief layout. The keyboard layout can be modified at any time.
- Unlimited keystroke macros. Allows single-key access to menu functions and creation of custom editing functions. Keystroke macros may be built into VEDIT, saved to and loaded from disk.
- Fully adjustable tab stops. The <Tab> key can enter a tab character or, optionally, spaces to the next tab position. Configurable tab-fill, “newline” and “null” display characters.
- Word processor emulation. VEDIT can closely emulate other editors and word processors. Contains options for emulating common cursor positioning modes, other block operations, various styles of searching, functions performed by the <Tab> and <Enter> keys and much more.
- Totally configurable with over 200 parameters. VEDIT can be configured using menus or with easily edited **`vedit.cfg`** and **`vedit.key`** files.

Ready-To-Use Macros

VEDIT runs most “macros” written in the VEDIT PLUS programming language. Although the VEDIT manual does not document this programming language, many useful macros are supplied with VEDIT and can be used without understanding their internal workings.

These macros are easy to use and are fully documented in Chapter 5. Most of them can be run from the **{MISC}** or **{MISC, More macros}** menus. You can also create Windows icons or DOS batch files to automatically run macros when VEDIT is started.

The following macros are supplied with VEDIT:

- CFUNC - This C program outlining macro lists each C program routine declaration in a separate window; as you move through the list, the original window moves through the C program.
- PFUNC - Similar macro to CFUNC, but for Pascal programs.
- COMPARE - This file comparison macro displays the differences between two files on the screen and lets you merge or edit them in any way desired. After displaying a difference, the macro can re-align the files and 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.
- COMPDIR - This directory comparison macro compares all files in two directories and displays a list of which files are different. It also lists those files that are in one directory and not in the other.
- COMPILE - The integrated compiler support macro is described below.
- DISPLAY - This simple file display macro permits a computer user with no VEDIT experience to browse through a file. (It is used during the automated installation.)
- CTAGS - The “ctags” facility is useful when working on a program with many files. Once setup, placing the cursor on any function name and pressing a hot-key opens the file in which the function is declared, with the cursor on the function declaration.
- PRINT - This simple print formatting macro can be accessed from the normal Printing dialog box or can immediately be executed when VEDIT starts up. It adds the filename, date and page number at the top of each page. VEDIT PLUS users can enhance it for fancier print formatting.
- SORT - Sorts multi-line records, such as a typical mailing list of names and addresses. (Other sorting is built into VEDIT.)
- WILDFILE - This very useful 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 “*”. These files will be searched in the current directory and, optionally, in all subdirectories. (WILDFILE performs a superset of the UNIX “grep” command.)

Custom Macros and {USER} Menu

VEDIT comes with a library of additional “keystroke macros” that can be built into the editor and assigned to a “hot-key”.

- KEY-MAC.LIB - 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.

A set of custom editing functions can be added to the main menu; they appear in the special **{USER}** menu. The default **{USER}** menu includes some popular functions from KEY-MAC.LIB. As with all menu functions, the custom functions can have hot-keys assigned to them.

The editing functions in the **{USER}** menu are implemented using the VEDIT PLUS macro language. Although VEDIT PLUS is needed to fully realize its potential, VEDIT users can modify the existing functions or add new ones by copying macros from KEY-MAC.LIB.

Integrated Compiler Support

The integrated compiler support lets you run popular compilers, assemblers, linkers, debuggers and Make programs from within VEDIT. If compilation errors occur, VEDIT automatically loads the correct source file and positions the cursor on the error. You can then make corrections and press a key to move to the next error, or recompile.

The compiler support automatically adjusts to the needs of different programs in different directories, e.g. one program may be compiled as “large model” while another is compiled as “small model”. Compiler and other options can also be changed easily via menus.

The compiler support works with “make” programs and “makefiles” to further automate program development. It even tracks compilation errors reported by different compilers run within one Make script.

The following “make” programs are supported:

- Microsoft NMAKE
- Borland MAKE
- Avocet MAKE
- Other UNIX style Make programs

VEDIT supports these and other compilers:

- Microsoft: C/C++, Quick C, Quick Basic, Basic Compiler, Fortran, Cobol, MASM Assembler
- Borland: C++, Turbo C, Turbo Assembler, Turbo Pascal
- Sun: Java SDK
- Lahey: Fortran 77, Personal Fortran 77
- Metaware: High C
- Micro Focus: Cobol
- Microrim: Rbase Compiler
- Nantucket: Clipper Compiler
- Realia: Cobol
- SDS C/C++ CrossCode Compiler
- Watcom: C/C++ Compiler
- Whitesmith: 68HC11 C Compiler
- Zortech: C++
- Others: 2500 A.D., Acucobol, Avocet, Microtec, SVS, etc.

We can assist customers in adding support for new and custom compilers.

V-SPELL

V-SPELL Spelling Corrector

This is the ultimate spelling corrector that proofreads 50 pages (100 Kbytes) of text in under 10 seconds, displays misspelled words in context and can instantly suggest corrections for any misspelling. Since it automatically corrects the words in the document, there is no need to go back to the word processor. The 70,000 word main dictionary is expandable — you can have your own supplemental dictionaries and/or merge them into the main dictionary. You can even create your own main dictionary. Not only VEDIT, but most popular word processors are supported.

For typesetting, desktop publishing and other applications, V-SPELL can perform file hyphenation by inserting “soft hyphens” in all words of a document. Since the hyphenation points are built into the dictionary, you can be sure it is 100% correct — no error-prone algorithms are used! List price $79. Registered VEDIT users may purchase V-SPELL with manual for $49 or without manual for $25. (The text of the manual is on disk.)

System Requirements

The Microsoft Windows version requires:

- Windows 95/98/ME or Windows NT 4.0/2000/XP or later.
- VEDIT PLUS works well with minimal memory.
- 3 Megabytes of free disk space.

The DOS version requires:

- MS-DOS or PCDOS version 3.1 or later, or a compatible DOS such as OS/2 or DR DOS. The DOS version works very well under Windows. Windows 95/98/ME long filenames and OS/2 extended attributes are supported.
- An IBM PC compatible computer with an 80386 or better processor.
- At least 640 Kbytes of memory.
- A hard disk is highly recommended, but VEDIT can be installed on, and will run from, a floppy disk. A full installation with the compiler support requires about 2 Megabytes of disk space.
- A monochrome, CGA, MCGA, EGA, VGA or compatible display adapter and display.
- A special “Telnet” MS-DOS version supports most CRT terminals on the market including terminals externally connected to an IBM PC via Telnet, serial ports and/or modems.

Using this Manual

This manual assumes that you have a working knowledge of your computer and its basic operation. It is organized into the following chapters:

Introduction (Chapter 1)

Introduces VEDIT and lists the main features and system requirements.

Getting Started (Chapter 2)

Explains how to install VEDIT and make sure that it is working correctly. It lists the files on disk and the "Normal" keyboard layout.

Quick Tutorial (Chapter 3)

This short tutorial familiarizes you with the basic aspects of using VEDIT. It describes useful "shortcuts" and special features unique to VEDIT.

Editing Guide (Chapter 4)

This chapter covers the operation and main features of VEDIT in detail. It is the most important chapter and we hope you will find time to read it.

Advanced Topics (Chapter 5)

The startup macro **`startup.vdm`**, **{USER}** menu, color syntax highlighting and template editing are covered in detail. Describes how to use the "command macros" supplied with VEDIT.

Menu Reference (Chapter 6)

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 7)

Describes each basic edit function in detail.

Configuration (Chapter 8)

For the most part, you can configure VEDIT with the **{CONFIG}** menu. Alternatively you can change the keyboard configuration by directly editing the **`vedit.key`** and **`vedit.cfg`** files described in this chapter.

Messages (Chapter 9)

Lists and explains all error messages and common prompting messages.

Appendices

Topics include technical descriptions of VEDIT's file handling, network support and memory management.

Also includes a summary of search pattern matching, regular expressions, and additional keyboard layout information.

Troubleshooting.

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 key, such as **<Alt-E>**, which is typed by holding down the "**Alt**" key and typing the letter, in this case "**E**". "**Alt**" keys are frequently used to directly access menus, in this case the **{EDIT}** menu.
<Alt-Bksp>	This is an abbreviation for **<Alt-Backspace>**. Other abbreviations used are **<Ctrl-Bksp>** and **<Shft-Bksp>**.
<Enter>	The "Enter" key. Also labeled "Return" or "CR" on some keyboards.
<*key*>	Any other individual key such as **<Esc>**, **<Backspace>** , **<Space bar>** or **<F2>**.
<F1>	The default key for the **[HELP]** function. The actual key is displayed on the left side of the status line.
<F10>	The default key for accessing the main menu.
[*function*]	A basic editing function such as **[CURSOR UP]** or **[ERASE LINE]**. Each edit function name is usually followed with the keypress corresponding to the "Normal" keyboard layout. However, the keyboard layout is user configurable. The actual key assignments are displayed by selecting **{HELP, Keyboard layout}**.
{*menu, menu-item*} *x*	Menu system selections are shown within braces optionally followed by "***x***" to indicate variable parameters. The main menu is selected with the **[MENU]** function (**<F10>**). For example, **{GOTO, Line #} 71** denotes that you select "**Goto**" from the main menu, select "**Line #**" from the "**Goto**" menu, and then enter the value "**71**" at the line number prompt.
File_Open(*"file"*)	Describes the prototype for a macro language command. Items in *italics* are arguments.
File_Open("myfile.txt")	An example of an actual command. Commands can be entered in any combination of upper and lower case. The "_" is optional and is used to improve readability. Our convention is to capitalize each word of a command.
Search("|D|D")	The **Search( )** command uses pattern matching. The "|" is the **<Shift-\>** keyboard character. (This character is also used as the "OR" operator and is sometimes called the "pipe" character or "vertical bar".)

Product Support

Greenview Data provides free technical support for VEDIT to all registered users for 3 months. Should you have any problems that are not covered in this manual, please contact us, preferably by e-mail (support@vedit.com). Or call us. Please provide the following information when requesting support:

- VEDIT's exact version number. This is displayed on the status line when VEDIT starts up. It is also available in the main menu under **{HELP, About}**. We need both the ***version number*** and following ***date***.
- Operating system and version number. Particularly whether it is Windows 95/98/ME, Windows NT/2000/XP or DOS.
- We will often ask for copies of your VEDIT.CFG, VEDIT.KEY, STARTUP.VDM and USTARTUP.VDM files. You can "zip" them up and attach them to the e-mail message.

Web site: http://www.vedit.com

The following services are currently available on our extensive Web site:

- Free updates. During the 3 months of support, you can download newer versions. After the 3 months, you can purchase individual updates or the 18-month Software Subscription Plan.
- User Discussion Conference. Interact and share ideas and macros with other users.
- FAQ - the "Frequently Asked Questions" section contains common technical support questions and their answers.
- Detailed list of all enhancements and bug fixes. This helps you decide if an update is worthwhile.
- News and general marketing/sales information about VEDIT.

Return the Registration Card!

Registering your copy of VEDIT helps us support you. Please take a moment to fill out the enclosed registration card and drop it in the mail. You can also register on-line at www.vedit.com/register.htm.

Registered users receive these benefits:

- Technical support for 3 months by mail, telephone or FAX. Or send e-mail to support@vedit.com. Most questions are answered within 24 hours.
- Newsletters to keep you abreast of added features, new releases and helpful hints from other users.
- Discounts on future releases.

We welcome your comments and suggestions.

Chapter 2

Getting Started

Installation

The exact steps for installation depend upon which version of VEDIT you have — Windows or DOS.

NOTES: If you received a damaged CD, please contact us directly (and not your dealer). You can download the software from our Web site. Your serial number will activate the downloaded software as the full product.

Be sure to read the file README.TXT (DOS: README.BAT) before installing VEDIT. This file may contain last minute installation instructions.

Refer to Appendix D (Troubleshooting) if you have any trouble running VEDIT.

The Windows and DOS versions are supplied with an automated installation procedure that is both easy and flexible to use.

If you wish to install both the Windows and DOS versions, you must follow each installation procedure. We highly recommend installing both versions into the same directory; they will then have the same keyboard layout and configuration.

The automated installation performs the following operations:

- Installs the VEDIT files in the default **c:\vedit** or any other desired drive and directory.
- Selects whether the optional compiler support files are installed.
- Selects the initial keyboard layout, color scheme and other common configuration parameters.

  -OR- If a previous VEDIT is found, optionally transfers the current keyboard layout and configuration to the new version. Alternatively, the previous version can be archived (moved) to another directory.
- The Windows version creates the “VEDIT” program group and creates the initial icons in it. It creates the file **`vedit.ini`** in the VEDIT directory.

The Windows version is optionally installed into the Windows "Registry". This is highly recommended, but optional. It allows file types to be associated with VEDIT.

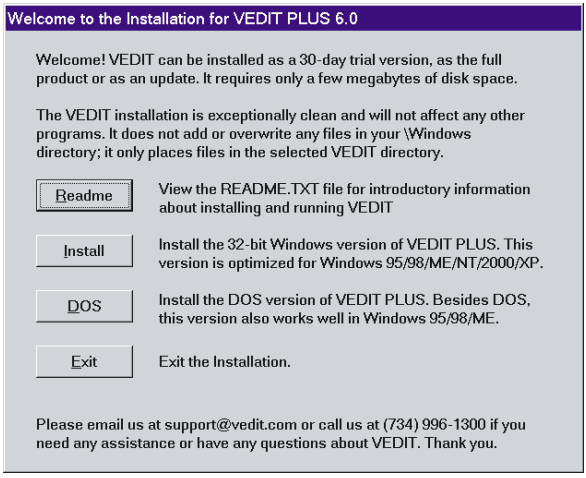
- The DOS version configures VEDIT to the specific operating system being used: DOS, Windows 3.1, Windows 95/98/ME, Windows NT/2000/XP or OS/2.

Once installed, VEDIT can immediately be used; there is no need to reboot the computer.

GOOD NEWS: The VEDIT installation does not modify your CONFIG.SYS, AUTOEXEC.BAT, WIN.INI or any other system files. Although we recommend installing VEDIT into the Windows registry, even that is optional and is only needed to associate file types with VEDIT. Also, since VEDIT does not use any .DLL files, installing (or un-installing) it will not affect other programs.

QNX: The Supplemental manual supplied with the QNX version of VEDIT PLUS describes the installation and configuration in detail. If this supplement is not available, first install VEDIT PLUS with the command "**/etc/install /dev/fd0**". Then print the supplemental text file **`/qnx4/vedit/readme.doc`**.

Windows Installation



NOTE: Be sure to read the file README.TXT before installing VEDIT. This file may contain last minute installation instructions.

➢ **To install the Windows version of VEDIT:**

1. Insert the VEDIT CD-ROM into your computer. The VEDIT installation will normally start automatically. If it does not start automatically, navigate to the "setup.exe" file on the CD-ROM and double-click it.
2. A pop-up dialog box will inform you that you are about to install VEDIT. Select **[Ok]** to continue.
3. VEDIT will then start! Instead of using a traditional installation program, VEDIT installs itself using a very flexible "macro".
4. You are then given the option of performing a detailed [Product] installation or a [Quick] installation. [Product] explains the installation step-by-step and gives you many options. [Quick] uses the default options and lets you install (or update) with a minimum number of steps.
5. One of the first selections is the destination directory into which to install VEDIT. This will later be referred to as the *VEDIT Home Directory*.

   The default is **c:\vedit**, but you can select any other directory such as **c:\program files\vedit**. If you are installing both the Windows and DOS versions, we highly suggest installing them both into **c:\vedit**. This will later be referred to as the *VEDIT Home Directory*.

   If this directory already exists, the installation assumes it contains a previous version of VEDIT. You are then given the option of archiving the entire directory by moving it to another directory, transferring the configuration from the previous VEDIT to the new one, or simply overwriting the previous VEDIT. Even if you choose overwriting, all configuration related files are first copied to the .\OLD subdirectory.
6. To activate VEDIT as the full product you should enter the serial-registration number when the installation prompts you. If you purchased VEDIT as a download, you should have received the serial number by email. If you purchased VEDIT with printed manuals, the serial number should be printed on the CD-ROM envelope. Otherwise, you can enter the serial number later by selecting **{HELP, Register VEDIT}.**
7. Follow the remaining prompts on the screen. It is easy to later change any configurations that you now select, such as the color scheme, without having to re-install.

We highly suggest selecting the "Normal" keyboard layout which closely follows Microsoft conventions. Otherwise, commonly used VEDIT functions will often not be accessible via a "hot-key".

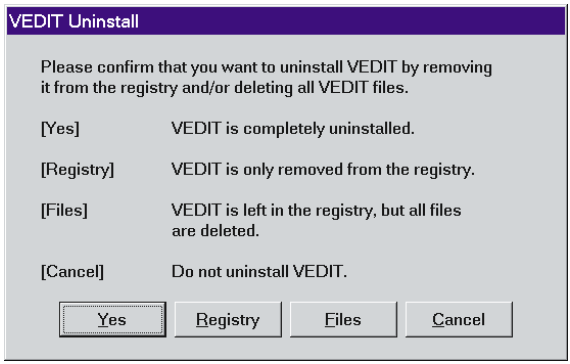
Skip down to the topic "Initial Configuration" below. Please also read the "Keyboard Layout" topic before using VEDIT.

Un-installing VEDIT

If VEDIT was installed into the Windows Registry, you can select "Add/Remove Programs" in the Control Panel to uninstall VEDIT. (This is the normal way to uninstall Windows programs.)

Alternatively, you can select the "Uninstall VEDIT" icon from the VEDIT Program group.

VEDIT will then startup and give you the following un-install options:



Besides completely uninstalling VEDIT, experienced users may find it useful to be able to only remove VEDIT from the Windows registry, or only remove (delete) the VEDIT files.

For example, if you think that VEDIT has somehow gotten corrupted, you may want to uninstall and re-install VEDIT. By leaving VEDIT in the registry, your file associations will not be lost.

It is also easy to manually uninstall VEDIT, e.g. the DOS version.

➢ **To manually remove (uninstall) VEDIT:**

1. Delete all files from the directory into which you installed VEDIT, by default **c:\vedit**. Also delete all subdirectories.
2. Remove the VEDIT directory, e.g. **c:\vedit**.
3. Delete all icons from the "VEDIT" program group; then delete the group.
4. If you configured VEDIT to create backup files by copying the original file to a backup directory, e.g. **c:\backup**, you may wish to delete this directory too.

DOS Installation

NOTES: Be sure to refer to the file README.BAT before installing VEDIT — give the DOS command **readme**. This file may contain last minute installation instructions.

The DOS version is fully compatible with MS-DOS or PCDOS 3.0 or later, DR-DOS and OS/2. It also works very well under Windows 95/98/ME and has long filename support. Although it will run, we do not recommend using it with Windows NT/2000/XP; for one thing it does not have long filename support under these OS.

Be sure to read the following sections in this chapter: "CONFIG.SYS and AUTOEXEC.BAT Files", "Path Command", "VEDPATH Environment Variable", "V-SWAP Installation" and "Hardware Configuration".

DOS Automated Installation

1. Insert the VEDIT CD-ROM into your computer. Start the installation by running the "install.bat" file which is in the root of the CD-ROM. For example, if your CD-ROM is drive "d:" and you are running from a DOS prompt, give the following commands:

   d: <Enter>

   install <Enter>

2. VEDIT will then start! Instead of using a traditional installation program, VEDIT installs itself using a very flexible "macro".

3. Continue with Step 4. above under "**Windows Installation**".

Skip down to the topic "CONFIG.SYS and AUTOEXEC.BAT Files" for further important installation instructions.

Then refer to the topic "Testing your Installation" below.

Hardware Configuration (DOS Version)

In order for the DOS version of VEDIT to run reliably and satisfy your needs, you should be aware of several important configuration options:

- Setting the Keyboard Typematic Rate
- Disabling Keyboard Polling for Windows NT and OS/2

Setting the Keyboard Typematic Rate

The rate at which the keyboard repeats keys when you hold them down is called the "typematic rate". The default rate of about 10 keys per second is slow for editing and should be speeded up.

VEDIT can optionally speed up the keyboard typematic rate inside VEDIT. However, this interferes with a faster rate you may have already set. It also causes problems with some machines resulting in slow startup, slow exiting or a "frozen" keyboard on exit. To disable having VEDIT change the typematic rate, set **{CONFIG, Misc, Keyboard typematic rate}** to "0". Then select **{CONFIG, Misc, Save into VEDIT.EXE}**.

NOTE: The DOS automated installation disables the features (sets it to "0") if you specify that your Operating System (OS) is Windows 95/98/ME/NT/2000/XP.

With DOS 5.0 and later and OS/2 we highly recommend that you set VEDIT's "Keyboard typematic rate" to "0" and instead set the desired typematic rate in your OS.

The DOS command "MODE CON RATE=*xx* DELAY=2" will set the rate to '*xx*'. You could add the following command to your AUTOEXEC.BAT file to set a very fast typematic rate:

```
MODE CON RATE=32 DELAY=2
```

Disabling Keyboard Polling (Technical, DOS only)

VEDIT normally polls the keyboard constantly. This is the most compatible mode with DOS and won't interfere with other programs. However it wastes CPU resource in a multi-tasking environment such as Windows.

Setting **{CONFIG, Misc, Keyboard polling}** to "1" turns off the constant polling. This is desirable with Windows and OS/2. However, it may cause conflicts with other programs and may not work on some systems.

Technical: To operate without polling, VEDIT must "hook" various hardware interrupts and set up mouse and timer handlers that directly modify the BIOS keyboard buffer. This can cause conflicts with memory-resident programs (TSRs).

The DOS automated installation disables keyboard polling (sets it to "1") if you specify that your Operating System (OS) is Windows or OS/2.

Installing DOS VEDIT in Windows

The DOS version of VEDIT is designed to work very well with Microsoft Windows. It can be installed with the supplied VEDIT.ICO icon.

Installing DOS VEDIT in Windows 95/98/ME

➢ **To install DOS VEDIT icon as a Windows 95/98/ME "shortcut":**

1. From Explorer or the Desktop, select the folder (group) to which to add VEDIT's icon. It can also be added to the top-level Desktop.
2. Right-click in any empty area of the folder or desktop and select "New"; then select "Shortcut".
3. The "Create shortcut" wizard will then prompt you.

   For "Command Line:", enter the full pathname to the VEDIT.EXE file. E.g. enter "**c:\vedit\vedit.exe**".

   For "Select a name for the shortcut", enter the program name you want to see under the icon, e.g. "VEDIT (DOS)".

   For "Select an icon", just press **[Finish]**; the VEDIT icon cannot be selected from here.
4. Right-click the new icon and select "Properties". Then select the "Program" tab.

   Be sure "Close on exit" is not enabled.
5. Select **[Change Icon...]**.

   For "File name:" enter the full pathname to the VEDIT.ICO file, e.g. **c:\vedit\vedit.ico**. Select the **[Ok]** buttons twice.

Notes:

From within VEDIT, press **<Alt-Enter>** to switch VEDIT between a full screen application and a windowed application. It works well in either mode.

The topic "Changing the VEDIT Icon Properties" in Chapter 4 describes how to set any desired startup invocation options.

Installing DOS VEDIT in Windows 3.1

VEDIT runs best from within Windows 3.x with the supplied VEDIT.PIF file. A ".PIF" file contains parameters on how to optimally run a program, e.g. how much memory to give it. Experienced users can use the Windows "PIF Editor" to change any desired parameters.

➢ **To install VEDIT into Program Manager with an icon:**

1. From inside Microsoft Windows, select the "Program Manager".
2. Select the Group to which you want to add the VEDIT icon. The "Main" group is a good choice.
3. Select the menu item **{FILE, New}**. Then select "Program Item".

4. Fill in the dialog box. For "Description," enter the program name you want to see under the icon, e.g. "VEDIT".

   For "Command Line", enter the full pathname to the VEDIT.PIF file. E.g. enter "**c:\vedit\vedit.pif**".

   For "Working Directory", enter the full pathname of the *VEDIT Home Directory*, e.g "**c:\vedit**".

   Leave "Shortcut Key" set to "None".

5. Select the **[Change Icon]** button. Windows warns you that no icon is available for VEDIT. Select the **[Ok]** button.

6. For "File name", enter the full pathname to the VEDIT.ICO file, e.g. "c:\vedit\vedit.ico". VEDIT's icon should be displayed. Select **[Ok]**.

7. You should now be back at main dialog box. Select **[Ok]**.

   The "VEDIT" icon should be displayed in your current Windows group.

8. Select the "VEDIT" icon. VEDIT should start up in full screen mode.

Installing DOS VEDIT in IBM OS/2

VEDIT can be installed under OS/2 as a normal DOS application. It can be run from the "C>" prompt in a normal DOS box or from its own icon.

We suggest making two configuration changes to VEDIT for use with OS/2:

- Set **{CONFIG, Misc, Keyboard Polling}** to "1" to turn off VEDIT's constant polling of the keyboard.
- Set **{CONFIG, Misc, Keyboard typematic rate}** to "0" and set the desired typematic rate with OS/2's "Keyboard Setup".

Remember to select **{CONFIG, Misc, Save into VEDIT.EXE}** to make any configuration changes permanent.

The default OS/2 "DOS Settings" work quite well with VEDIT and none are critical. However, the following settings may help VEDIT run better:

Background_Execution	Should be turned "Off" unless you set **{CONFIG, Misc, Keyboard Polling}** to "1".
DOS_Files	Should be set to "80" or more since VEDIT needs three handles for each file being edited.
Idle Sensitivity	Setting this to "100" gives more CPU time to DOS programs at the expense of multi-tasking. Some DOS programs work dramatically better with a setting of "100"; VEDIT runs a little better. It probably doesn't matter if **{CONFIG, Misc, Keyboard Polling}** is set to "1".

VEDIT automatically detects when it is running as a DOS application within OS/2 (2.1 and later including Warp); it then maintains the "Extended Attributes" when editing OS/2 files. Please note that newly created files will not be given any extended attributes.

Testing the VEDIT DOS Installation

After installing VEDIT, start it up to make sure that everything is working.

1. Assuming that your PATH command includes the directory containing VEDIT, you can start up VEDIT from any directory by typing:

   vedit <Enter>

   VEDIT will start up and display its signon message on the bottom line. There should also be a (flashing) cursor in the upper-left corner.

2. Press the **<Esc>** key to display the **{ESCAPE}** menu in the middle of the screen.

   If you cannot read the menu, cannot tell which item is currently selected, or cannot tell which letter in each item is highlighted, your screen colors are not set correctly. If you have a monochrome (black and white) display, you may need to start up VEDIT with the "**-m**" option. Then refer to **{CONFIG, Colors}** in Chapter 6 (Menu Reference) on how to configure VEDIT for a monochrome display.

3. Press the **[HELP]** key. With the "Normal" keyboard layout it is **<F1>**. The key assigned to the **[HELP]** function is displayed on the status line.

   You should now see a "pop-up" window that displays help information about the **{ESCAPE}** menu.

   If you get the error message "FILE NOT FOUND:", VEDIT could not find its on-line help file **`vphelp.hlp`**. Change **{CONFIG, Misc, VEDIT Home Directory}** to the directory into which you installed VEDIT and which should contain **`vphelp.hlp`**. Then select **{CONFIG, Misc, Save into VEDIT.EXE}** to make the change permanent.

   Press **<Esc>** to exit the on-line help window.

4. Exit VEDIT by selecting "Exit" from the **{ESCAPE}** menu. You should now be back in DOS.

Refer to Appendix D (Troubleshooting) if you encounter any trouble running VEDIT.

If you encounter any problems with the keyboard or mouse, or VEDIT appears to crash on startup or exit, you should change the following configuration settings to their most compatible settings:

- Set **{CONFIG, Misc, Keyboard typematic rate}** to "0".
- Set **{CONFIG, Misc, Keyboard polling}** to "0".
- Set **{CONFIG, Misc, Mouse cursor}** to "65" or "66".

If necessary, you can force the first two configuration settings on startup with the "**-j**" and "**-k**" invocation options:

vedit -j -k

Then select **{CONFIG, Misc, Save into VEDIT.EXE}** to make these configuration settings permanent.

CONFIG.SYS and AUTOEXEC.BAT Files

This topic applies only if you are running:

- VEDIT for DOS.
- VEDIT for DOS under Windows 3.1.

With DOS and Windows 3.1, you should check the CONFIG.SYS file in the root directory of your hard (boot) disk. It should contain the two lines:

FILES=80

BUFFERS=10

It is OK if the numbers are larger than these!

To simultaneously edit as many files as possible (32), we recommend that your CONFIG.SYS file contain "FILES=99". VEDIT gives the error "TOO MANY FILES" if you attempt to edit more files than the "FILES=xx" statement allows.

The "BUFFERS=10" statement is not really necessary, but speeds up not only VEDIT, but virtually every other program. No DOS computer should be without it.

PATH Command

To run the DOS VEDIT, DOS must know where to find VEDIT.EXE. We suggest simply copying VEDIT.EXE to a directory, typically "\DOS" or "\BIN", that DOS already searches for its other programs. Alternatively, add the *VEDIT Home Directory*, e.g. "C:\VEDIT", to the PATH command in your AUTOEXEC.BAT file.

V-SWAP Installation

We highly recommend that all DOS VEDIT users install the supplied V-SWAP program via their AUTOEXEC.BAT file. V-SWAP works in conjunction with VEDIT to swap out not only the editor but any desired memory resident programs (TSRs) when running DOS programs from within VEDIT.

Although V-SWAP is primarily intended for running compilers from within VEDIT, almost all VEDIT users will find V-SWAP useful. V-SWAP is very small and unlikely to cause conflicts with other programs.

The on-line help topic "VSWAP" describes V-SWAP and its installation in detail, including how to use V-SWAP with Microsoft Windows (tm).

Network Installation

Customers that have purchased two or more licenses of VEDIT (such as the 5-User Pack) can simply install VEDIT on each licensed user's workstation.

Alternatively, some network administrators may prefer to install VEDIT on the network server. This makes it easier to update the software and monitor license usage. The network installation is designed so that each user only needs "Read" access, and not "Read/Write" access, to the VEDIT files on the server.

When VEDIT is installed on a network server, each user can still have their own personal configuration and keyboard layout. Each user can also set up their own color syntax highlighting, template editing, a **{USER}** menu, etc.

➢ **To install VEDIT on a network server:**

1. Install VEDIT on the server, following the normal installation instructions. Let's assume it is installed into the directory "h:\apps\vedit". This will be the *VEDIT Home Directory*.

2. Create the same directory (folder) on each user's local hard disk. Let's assume it is **`c:\vedit`**. This will be the *User Config Directory*. It will only contain the **`startup.vdm`**, **`ustartup.vdm`**, **`vedit.cfg`** and **`vedit.key`** files. The Windows version will also contain **`vedit.ini`**.

3. Within the *User Config Directory* on each user's local hard disk, create a subdirectory (folder) called "**backup**", e.g. create the subdirectory **`c:\vedit\backup`**. Users can optionally configure VEDIT to save a backup copy of each edited file in this directory.

4. Within the *User Config Directory* on each user's local hard disk, create a subdirectory (folder) called "**temp**", e.g. create the subdirectory **`c:\vedit\temp`**. Some macros supplied with VEDIT use this directory.

5. Within the *User Config Directory* on each user's local hard disk, create a subdirectory (folder) called "**file-cfg**", e.g. create the subdirectory **`c:\vedit\file-cfg`**. This will be used for the file-open configuration feature (in the new VEDIT version 6.0).

   (VEDIT version 6.0) The \FILE-CFG subdirectory can be left empty, but it may help users get started if you copy the .CFT files from VEDIT's \FILE-CFG subdirectory on the server to each user's local hard disk. E.g. on each user's machine copy all files from **`h:\apps\vedit\file-cfg`** to **`c:\vedit\file-cfg`**.

6. Within the *User Config Directory* on each user's local hard disk, create a subdirectory (folder) called "**user-mac**", e.g. create the subdirectory **`c:\vedit\user-mac`**. Experienced users can save their own custom VEDIT macros here.

7. Edit the **`vedit.ini`** file on the server to reference the correct directories. For example, it should contain the lines:

   ```
   HomeDir=h:\apps\vedit
   MacroDir=h:\apps\vedit\macros
   BackupDir=c:\vedit\backup
   UserCfgDir=c:\vedit
   UserMacroDir=c:\vedit\user-mac
   FileCfgDir=c:\vedit\file-cfg
   VeditTempDir=c:\vedit\temp
   Startup=startup.vdm
   ```

8. Copy the modified **`vedit.ini`** file from the server to each user's local directory e.g. copy it to **`c:\vedit\vedit.ini`**.

9. Copy the **`startup.vdm`**, **`ustartup.vdm`**, **`vedit.cfg`** and **`vedit.key`** files from the server to each user's local directory e.g. copy them from **`h:\apps\vedit`** to **`c:\vedit`**.

   On startup, VEDIT looks for the **`startup.vdm`** file first in the "current" directory, then in the *User Config DirectoryI, and last in the VEDIT Home DirectoryI.*

10. Create a VEDIT icon on each user's workstation. Set the VEDIT icon's properties "Target" to:

    ```
    h:\apps\vedit\vpw.exe
    ```

See also:

The topic "Configuration" in Chapter 8.

The topic "Startup.vdm File" in Chapter 4.

Initial Configuration

Just about every aspect of VEDIT's operation is configurable. As you become more familiar with VEDIT, you will probably configure VEDIT more and more to your needs and personal preferences. This manual describes configuration in three places:

- Most configuration changes are made with the **{CONFIG}** menu, as described in Chapter 6 (Menu Reference).

  The keyboard layout can be changed with the **{CONFIG, Keyboard layout}** sub-menu.

- The configuration and keyboard layout can alternatively be changed by directly editing the **`vedit.cfg`** and **`vedit.key`** files as described in Chapter 8 (Configuration). Experienced users often prefer this method.

- More advanced configuration topics such as the changing the **{USER}** menu and setting up color syntax highlighting are described in Chapter 5 (Advanced Topics).

  Optional "File-type specific configuration" configures the editor according to the filename extension for each file opened. VEDIT supports different configurations for each open file. For example, with a ".c" file, auto-indenting is enabled, while with a ".txt" file, word wrap is enabled. This is also covered in Chapter 5.

Here are some non-default configuration changes you may want to make soon.

- Set **{CONFIG, File handling, Auto-save interval}** to "15" to have VEDIT auto-save your changes every 15 minutes.

- Set **{CONFIG, Programming, File-type specific config}** to to "7" to configure VEDIT according to a file's type, e.g. filename extension.

- By default, VEDIT creates backup files by renaming the original file with a ".BAK" filename extension. Alternatively, set **{CONFIG, File handling, Backup files}** to "2" to create backup files by copying the original file to the *VEDIT Backup Directory*, by default **c:\vedit\backup**.

Saving Configuration Changes

By default, **{CONFIG, Auto-save config}** is enabled. Any changes you make in the **{CONFIG}** menus are then automatically saved for the next time you run VEDIT, just as if you had manually selected **{CONFIG, Save config}**.

As you become more experienced with VEDIT, you may want to disable **{CONFIG, Auto-save config}**. You can then make temporary configuration changes that are not automatically saved, i.e. they are lost when you exit VEDIT. To make changes permanent, you must then explicitly select **{CONFIG, Misc, Save config}**.

Description of Files

Windows Version

README.TXT	Last minute notes to read before getting started.
SETUP.EXE	Installation program. It starts up VEDIT with the INSTALLW.VDM macro to perform the actual installation and initial configuration.
VPW.EXE	Executable VEDIT (32-bit) for Windows 95/98/ME/NT/2000/XP.
INSTALLW.VDM	VEDIT macro that controls the automated installation.
VEDIT.INI	VEDIT for Windows file which stores startup and configuration information. It is fully commented.
VEDIT.FON	VEDIT font file for Windows provides attractive and useful fixed-width display fonts in various sizes in both the "ANSI" and "OEM" character sets.
VPW-FILS.CAB	The installation expands this file to create the Windows version specific files, in particular the on-line help file VEDITPW.HLP and the .PIF files.

The following files are placed into the VEDIT directory during installation by expanding the VPW-FILS.CAB file.

VEDITPW.HLP	The VEDIT for Windows on-line help file.
MSDOS95.PIF	A .PIF file used to shell out to a maximized (large) DOS box in Windows 95/98/ME with **{MISC, DOS Shell}**.
MSDOS95I.PIF	A .PIF file used to shell out to a minimized (invisible) DOS box in Windows 95/98/ME.
MSDOS95W.PIF	A .PIF file used to shell out to a windowed DOS box in Windows 95/98/ME with **{MISC, Run DOS program}**.
MSDOS95C.PIF	A .PIF file used to shell out to a windowed, self-closing DOS box in Windows 95/98/ME.
MSDOSNT*.PIF	Similar files for Windows NT/2000/XP.

Windows and DOS Version (Common Files)

VP-FILS.CAB	The installation expands this file to create many additional files, in particular the .VDM macro files.
VCS-FILS.CAB	The installation optionally expands this file to create the compiler support files in the .\COMPILE subdirectory.

TUTORFIL.CAB	The installation expands this file to create the tutorial files in the .\TUTOR subdirectory.
USER-MAC.CAB	The installation expands this file to create the user supplied macros placed in the .\USER-MAC subdirectory. ERRATA.TXT describes the files.
EBCDIC-T.CAB	The installation optionally expands this file which contains a trial version of our EBCDIC Level-2 conversion software. This can convert EBCDIC (mainframe) files with packed, signed, zoned and other special fields into ASCII. A COBOL "copy-book" can control the conversion. See the file EBCDIC-T.TXT (in the VEDIT directory) for details.
ERRATA.TXT	Contains errata to this manual and describes recent enhancements that are not documented in the manual. Please note that the on-line help is always completely up-to-date.
WHATSNEW.TXT	Description of new features added to this release of VEDIT; primarily oriented to previous VEDIT users.
NEW-CMDS.TXT	A summary of recently added, changed or enhanced macro language commands; primarily oriented to technical users.

The following files are placed into the VEDIT directory during installation by expanding VP-FILS.CAB.

KEY-MAC.LIB	Library of useful keystroke macros that can be added to VEDIT. View the beginning of this file for more information. The topic "Keystroke Macros" in Chapter 4 (Editing Guide) contains a step-by-step example.
BRIEF.KEY	Keyboard layout file to emulate Brief (tm). Described in BRIEF.DOC.
WORDSTAR.KEY	Keyboard layout file to emulate WordStar (tm). Described in WORDSTAR.DOC.
WORDPERF.KEY	(DOS only) Keyboard layout file to emulate WordPerfect (tm). Described in WORDPERF.DOC.
USER.MNU	The default **{USER}** menu; it is loaded by the STARTUP.VDM file.
TUTOR.MNU	The default **{TOOLS}** menu is used as the {TUTORIAL} menu; it is loaded by the STARTUP.VDM file.
ANSI.TBL	Translation table for converting between IBM PC Graphics characters and ANSI (Windows) graphics characters.
EBCDIC.TBL	Translation table for converting between the ASCII and EBCDIC (IBM mainframe) character sets. Note that EBCDIC.TBL is built into VEDIT.

USER.TBL	A prototype translation table that can be used to create a custom translation table.
*.SYN	Color syntax highlighting files are supplied for C, Clipper, Cobol, Folio, HTML, Java, MBasic, Pascal, Perl, Rexx, SQL, Systat and others. They can be manually loaded with **{MISC, Load syntax file}** or automatically loaded by the STARTUP.VDM file.
*.VTM	Template editing macros are supplied for C, HTML, Java and VEDIT. They can be manually loaded with **{MISC, Load template file}** or automatically loaded by the STARTUP.VDM file.
*.VDM	VEDIT is supplied with many macros which have a **.vdm** filename extension. Some are automatically run by menu functions while others are run from **{MISC, More macros}** or **{MISC, Load and execute macro}**. See the on-line help topic "Description of Files" (DOS: "FILES") for a description of most of the macros.
STARTUP.VDM	This special macro that is executed when VEDIT starts up. A copy of the original macro is supplied as STARTUP.ORG. Extensively described in Chapter 5.

DOS Version Only

READ-DOS.TXT	Last minute notes to read before getting started.
INSTALL.BAT	Batch file to start up the automated installation using VEDIT.EXE and INSTALL.VDM.
INSTALL.VDM	VEDIT macro that controls the automated installation.
EXTRACT.EXE	The program to expand (similar to "unzipping") the .CAB files. It is used only during installation.
VEDIT.EXE	Executable VEDIT program for DOS.
VEDIT.ICO	"Icon" file for installing the DOS version of VEDIT into Windows.
VEDIT.PIF	"Program Information File" for running the DOS version of VEDIT under Microsoft Windows 3.x.
VPD-FILS.CAB	The installation expands this file to create the DOS version specific files, in particular the on-line help file VPHELP.HLP.

The following files are placed into the VEDIT directory during installation by expanding the VPD-FILS.CAB file.

VPHELP.HLP	VEDIT PLUS for DOS on-line help file.
P.BAT	Batch program to start up the PRINT.VDM macro.
VV.BAT	A batch file for loading V-SWAP and running VEDIT. See the on-line help topic "VSWAP" for details.

WILD.BAT	Batch file to start up the WILDFILE.VDM macro.
VGA34.COM	Tiny program that puts a VGA into 34 line mode by switching into 480 scan-line mode. This may also reduce flicker on some monitors. Use the DOS "mode co80" command to return to normal 25 line mode.
VSWAP.EXE	Program for swapping the DOS version of VEDIT out of memory during a "DOS Shell".
VPLUSOS2.ICO	VEDIT PLUS icon for OS/2.

Compiler Support File Names

The following files make up VEDIT's Compiler Support. See "Integrated Compiler Support" in Chapter 5 for more information.

COMPILE.VDM	This macro is executed by **{MISC, Load compiler support}** to load the compiler support items into the **{TOOLS}** menu.
COMPILE.CNF	Configuration file that is edited by the user to specify the default Compiler, Linker, Debugger and Make commands.
COMPILE.MNU	This file sets up the Compiler support items in the **{TOOLS}** menu. It can be loaded by **{MISC, Load compiler support}** or by the STARTUP.VDM macro.
C-*.VDM	These macro implement each of the compiler support items in the **{TOOLS}** menu. These files reside in the .\COMPILE subdirectory.
JAVA-SDK.CNF	Configuration file that is edited by the user to specify the default Java SDK commands.
JAVA-SDK.MNU	This file sets up the Compiler support items in the **{JavaTools}** menu. It can be loaded by **{MISC, Load compiler support}** or by the STARTUP.VDM macro.
*.VCS	Compiler specific macros; there is a file for each supported compiler. These files reside in the .\COMPILE subdirectory.
GENERIC.VCS	Generic macro for running any compiler. It only displays the compiler's output, it does not automatically track errors.
SAMPLE.VCS	This sample compiler specific macro is heavily commented and can be used as a model for developing a custom support for other compilers.

Keyboard Layout

VEDIT's keyboard layout is completely configurable. The automated installation lets you select the initial keyboard layout. You can later modify the initial layout with **{CONFIG, Keyboard layout, Edit/view layout}** or by directly editing the VEDIT.KEY file.

Unless you have a very strong personal preference, we recommend you start with the "Normal" keyboard layout. It is very compatible with modern Microsoft and other Windows programs. Additional keys have been carefully chosen to work well with VEDIT's special features.

All examples in this manual list the name of the edit function or menu item and the corresponding keystroke in the "Normal" keyboard layout. Examples are: **[HELP]** (**<F1>**) and **{FILE, Print}** (**<Ctrl-P>**).

The initial Windows and DOS keyboard layouts are:

- NORMAL. The recommended layout.
- WORDSTAR. Emulates WordStar using control, cursor and function keys. It is fully described in the file WORDSTAR.DOC.
- WORDPERF. (DOS Version Only) Emulates Word Perfect. Since it has few menu "hot-keys", it helps to access menu functions with a mouse. It is fully described in the file WORDPERF.DOC. It is incompatible with Windows conventions and therefore not supplied with the Windows version.
- BRIEF. Emulates the Brief editor. It is fully described in the file BRIEF.DOC.

Why <Ctrl-S> is different in VEDIT

Most Windows programs assign **<Ctrl-S>** to **{FILE, Save}**, but VEDIT does not. By default, the key is unassigned because it is too easy to press by mistake.

Our experience is that you may be making temporary editing changes during a complicated "cut and paste" operation that you don't want to save. Also, when editing a huge multi-megabyte file, an unneeded file save can waste several minutes. (It take several minutes to copy a 100 Megabyte file and it takes VEDIT just as long to save the file.)

Sometimes saving unwanted editing changes can be disastrous. For the same reason, VEDIT's "auto file save" feature is disabled by default.

If you don't like our reasons, you can assign **<Ctrl-S>** to **{FILE, Save}**. (See "Keyboard Layout" in the index.)

Otherwise, you may want to assign **<Ctrl-S>** to **{GOTO, Set marker}** in place of the default **<Ctrl-D>**.

Notes on the "Normal" Layout

- The "**<Numpad>**" keys are on the very right-hand side on the IBM PC keyboard. VEDIT treats the **<Numpad>** keys as function keys.

  If desired, they can be assigned as keystroke macros to the displayable characters "+", "-", "*" and "/".

- The "Normal" layout redundantly assigns **[T-REG INSERT]** to **<Numpad*>** and **<F11>**, **[T-REG COPY]** to **<Numpad+>** and **<CTRL-F11>**, and **[T-REG MOVE]** to **<Numpad->** and **<Alt-F11>**. Although we suggest using the **<Numpad>** keys, many laptop computers do not have **<Numpad>** keys, but do have **<F11>**. If desired, you can un-assign one set of keys; this is described under "Keystroke Macros" in Chapter 4 (Editing Guide).

NOTE: The following pages list the "Normal" layout in alphabetic order by function name and the keystroke macros in alphabetic order by key name. Appendix E lists the entire layout in alphabetic order by key name. The layout is available on-line via **{HELP, Keyboard layout}** and **{CONFIG, Keyboard layout, Edit/view layout}** , from which you can also print the layout. The status line always displays the current key assigned to the **[HELP]** function.

“Normal” IBM PC Keyboard Layout

[RETURN	**<Enter>**		
[BACKSPACE]	**<Backspace>**		
[BACKTAB]	**<Shft-Tab>**		
[CANCEL]	**<Ctrl-\>**	or	**<Ctrl-Break>**
[CURSOR UP]	**<Up Arrow>**		
[CURSOR DOWN]	**<Down Arrow>**		
[CURSOR RIGHT]	**<Right Arrow>**		
[CURSOR LEFT]	**<Left Arrow>**		
[DELETE]	**<Del>**		
[DEL PREV WORD]	**<Ctrl-Bksp>**		
[DEL NEXT WORD]	**<Ctrl-Del>**		
[ENTER CTRL]	**<Ctrl-Q>**	or	**<Ctrl-Shift-^>**
[ERASE BOL]	**<Ctrl-J>**		
[ERASE EOL]	**<Ctrl-K>**		
[ERASE LINE]	**<Ctrl-L>**		
[ESCAPE]	**<Esc>**		
[HELP]	**<F1>**		
[INSERT TOGGLE]	**<Ins>**		
[LINE BEGIN]	**<Home>**		
[LINE END]	**<End>**		
[MENU]	**<F10>**	or just tap	**<Alt>**
[NEXT LINE]	**<Ctrl-Enter>**		
[NEXT PARAGRAPH]	**<Ctrl-Down Arrow>**		
[NEXT TAB STOP]	**Not assigned**		
[NEXT WORD]	**<Ctrl-Right Arrow>**		
[PAGE UP]	**<PgUp>**		
[PAGE DOWN]	**<PgDn>**		
[PREV PARAGRAPH]	**<Ctrl-Up Arrow>**		
[PREV WORD]	**<Ctrl-Left Arrow>**		
[REPEAT]	**<Ctrl-R>**		
[REPEAT LAST]	**<Alt-R>**		
[SCREEN BEGIN]	**<Ctrl-PgUp>**		
[SCREEN END]	**<Ctrl-PgDn>**		
[SCROLL UP]	**<Alt-Up Arrow>**		
[SCROLL DOWN]	**<Alt-Down Arrow>**		
[SCROLL RIGHT]	**<Alt-Right Arrow>**		
[SCROLL LEFT]	**<Alt-Left Arrow>**		
[TAB CHARACTER]	**<Tab>**		
[T-REG COPY]	**<Numpad+>**	or	**<Ctrl-F11>**
[T-REG MOVE]	**<Numpad->**	or	**<Alt-F11>**
[T-REG INSERT]	**<Numpad*>**	or	**<F11>**
[VISUAL ESCAPE]	**<Alt-F10>**		
[VISUAL EXIT]	**<Ctrl-E>**	or	**<Ctrl-F10>**

"Normal" Built-in Keystroke Macros (Hot-keys)

<Alt-A>	Add a new keystroke macro
<Alt-D>	Toggle current window through display modes
<Alt-I>	Set a "column" type block marker
<Alt-J>	Toggle between window colors
<Alt-K>	Start/stop recording a new keystroke macro
<Alt-L>	Set a "line" type block marker
<Alt-O>	Open a new file in same (current) buffer
<Alt-Q>	Toggle File-selector window on and off.
<Alt-X>	Exit VEDIT. Same as **<Alt-F4>**.
<Alt-Y>	Open another file in a horizontal window
<Alt-Z>	Zoom or de-zoom windows
<Alt-=>	Toggle binary/text display mode
<Alt-\>	Toggle hex mode split
<Alt-/>	Create (remove) command mode window
<Alt-[>	Goto the beginning of the block
<Alt-]>	Goto the end of the block
<Alt-Bksp>	Undo last edit operation (keystroke)
<Alt-Enter>	Display file properties and VEDIT status
<Alt-F1>	Display the current basic keyboard layout
<Alt-F2>	Start a new search & replace
<Alt-F4>	Exit VEDIT - save/abandon file(s)
<Alt-F5>	Switch to the selected window (file)
<Alt-F9>	Highlight, move block to cursor position
<Alt-F12>	Save current file; continue editing
<Ctrl-A>	Select entire file as a block
<Ctrl-B>	Format the current paragraph
<Ctrl-C>	Copy to Windows clipboard
<Ctrl-D>	Set a text marker
<Ctrl-F>	Start a new search
<Ctrl-G>	Goto a text marker
<Ctrl-H>	Start a new search & replace
<Ctrl-I>	Start a new incremental search
<Ctrl-N>	Open a new (empty) buffer.
<Ctrl-O>	Open another file in new buffer; full window
<Ctrl-P>	Print current file or highlighted block
<Ctrl-U>	Undo current/previous line
<Ctrl-V>	Insert (paste) the Windows clipboard
<Ctrl-W>	Close current window
<Ctrl-X>	Move (cut) to Windows clipboard
<Ctrl-Y>	Redo last undo operation
<Ctrl-Z>	Undo last edit operation (keystroke)
<Ctrl-]>	Goto matching parenthesis - ( ) < > [ ] { }
<Ctrl-End>	Goto the end of the file
<Ctrl-F1>	Help - search for a keyword
<Ctrl-F2>	Start a new search of all open buffers (files)
<Ctrl-F3>	Compare any two open buffers (files)
<Ctrl-F4>	Close current window
<Ctrl-F5>	Toggle to the previous window
<Ctrl-F6>	Toggle to the next window

<Ctrl-F7>	Load a macro in a text register and execute it
<Ctrl-F8>	Execute the command macro in a text register
<Ctrl-F9>	Highlight, copy block to cursor position
<Ctrl-Home>	Goto the beginning of the file
<Ctrl-Ins>	Copy to the clipboard. Same as **<Ctrl-C>**.
<Ctrl-Shift-C>	Copy to register 0 (the "scratchpad")
<Ctrl-Shift-O>	Open another file; use simple (quick) prompt
<Ctrl-Shift-P>	Play back the quick macro
<Ctrl-Shift-R>	Record a quick macro
<Ctrl-Shift-V>	Insert (paste) register 0 (the "scratchpad")
<Ctrl-Shift-X>	Move (cut) to register 0 (the "scratchpad")
<Ctrl-Shift-Z>	Undo current/previous line
<Ctrl-Tab>	Toggle to next window; same as **<Ctrl-F6>**
<F2>	Start a new search
<F3>	Search/replace again for next occurrence
<F4>	Switch to the selected buffer (file)
<F5	Toggle to the previous buffer (file)
<F6>	Toggle to the next buffer (file)
<F7>	Undent left margin or highlighted block
<F8>	Indent left margin or highlighted block
<F9>	Set a "stream" type block marker
<F12>	Perform a "ctags" lookup of the current symbol. (See "Ctags Symbol Lookup" in Chapter 5 for more information.)
<Numpad+>	Copy block to selected text register
<Numpad->	Move (cut) block to selected text register
<Numpad*>	Insert (paste) selected text register
<Shft-Del>	Move (cut) to the clipboard. Same as **<Ctrl-X>**.
<Shft-F3>	Search/replace again for previous occurrence
<Shft-F9>	Remove (cancel) any block markers
<Shft-Ins>	Insert (paste) the clipboard. Same as **<Ctrl-V>**.

Optional Keys assigned in STARTUP.VDM

The **`startup.vdm`** file can optionally set up the following keystroke macros. See the comments in **`startup.vdm`** for details.

<Alt-0>	Open the file **`mynotes.txt`** for editing personal notes.
<Alt-F12>	Configure VEDIT to a xBase ".DBF" file.
<Ctrl-F11>	Run the V-SPELL spelling corrector.
<Numpad/>	Toggle **{CONFIG, Programming, Lower/Upper case key conversion}** on and off. Useful for assembly language programming.
<Numpad.Enter>	Alternative key for **[VISUAL EXIT]**. Useful with WILDFILE macro to search for next occurrence.

Chapter 3

Quick Tutorial

Let's take a quick tour of VEDIT. This chapter covers the basics of opening, editing, printing and saving a file. In short, you will know all that's required to really use VEDIT!

Each basic editing function in VEDIT has a name that we normally show such as **[HELP]** and **[CURSOR RIGHT]**. We also show which key to press to perform these editing functions: **<F1>** and **<Cursor Right>**.

To keep things simple, this quick tutorial often just lists which key to press to perform various editing operations. However, the rest of VEDIT's documentation and on-line help is oriented towards the names of the editing functions.

NOTE: Since this quick tutorial is specific on which keys to press, it assumes that you have installed the Windows or DOS version of VEDIT and selected the "Normal" keyboard layout.

If you notice something about VEDIT's behavior that you don't like, don't worry. As you will see, just about everything about VEDIT is configurable. For example, you can easily configure common keys like **<Enter>**, **<Tab>**, **<Begin>**, **<End>** and the cursor "arrows" to emulate any other editor.

Starting VEDIT

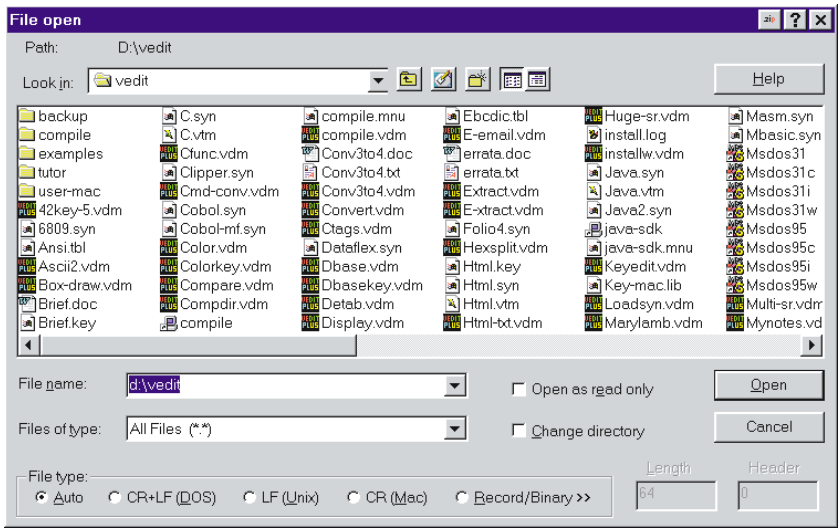
➢ **To start VEDIT and begin editing a file:**

1. Double-click the VEDIT icon with the mouse.
2. Select **{FILE, Open}**. This notation means that you select "File" from the main menu and then select the "Open" item.

   As a convenience, this function can be used both to open an existing file or to create a new file.

   Alternatively, just press **<Ctrl-O>** which is the "*hot-key*" for this function. The menu displays what the hot-keys are.

   Alternatively, press the "Open" icon on the toolbar.



3. You are now in the File-Open dialog box. At the "Filename:" prompt, enter the desired filename, e.g. "**news.txt**". If the file does not exist, it will be created.

   To edit an existing file, you can also use the typical Windows "Point & Shoot" to select the file.

   If possible, select an existing file which you can safely experiment with.

You can now edit the new or existing file as desired. Note the "*status line*" at the bottom of the screen.

Notes:

You can also open existing files by using "*drag and drop*" to drag the desired file(s) to the VEDIT icon, or onto a running VEDIT.

If the toolbar is not displayed, select **{VIEW, Toolbar}** to enable it.

Entering New Text

If you didn't select an existing file, you should enter one or two short paragraphs of text.

By default, VEDIT starts up in "Insert" mode — text you type is inserted at the cursor position instead of overstriking existing text. Note the "INS" on the status line. If you do not see "INS" on the status line, press **<Ins>** once. VEDIT is usually configured to have distinctive cursors in Insert Mode and Overstrike Mode.

When entering lines of text, press the **<Enter>** key at the end of each line. Pressing the **<Enter>** key moves the cursor to the beginning of the next line. Pressing **<Enter>** while in Insert mode, or at the end of the file, inserts a "newline" character. Every text line ends in a (invisible) "newline" character.

Therefore, pressing **<Enter>** in the middle of a line splits the line into two lines. (You must be in Insert mode.)

Now try typing a very long line. When you go past the right window edge, VEDIT will horizontally scroll the window to handle the long line.

To simplify entering text for word processing, you can enable "*word wrap*". VEDIT then automatically wraps words to the next line when you reach the right edge of the window or a configurable "*right margin*".

➢ **To enable "word wrap" to simplify word processing:**

1. Enable **{CONFIG, Word processing, Enable word wrap and formatting}**. This notation for configuration parameters means:

   1a. Select **{CONFIG}** menu and then the "Word processing" item.

   1b. Select the item "Enable word wrap and format paragraph".

   1c. Select the **[Close]** button to exit the configuration dialog box.

2. The default right margin is the edge of the window. Alternatively, you can select a specific right margin. For example, a right margin of "70" is often good for printing.

   If desired, select a specific right margin with **{CONFIG, Word processing, Right margin}**.

Now try entering a long line of text and notice how words are wrapped to the next line when you reach the right margin.

Unlike word processors, VEDIT never adds strange control characters to your file, even when using word wrap. VEDIT simply inserts the "newline" character to start a new line, just as if you had pressed **<Enter>**.

Notes:

The "newline" character is usually two characters - <CR> Carriage-Return and <LF> Line-Feed. Technically, it depends upon whether you are editing a DOS/Windows, Unix or Macintosh file. See "File Types" in Chapter 4 for details.

The "newline" character(s) is normally invisible, but you can configure it to be visible with **{CONFIG, Characters/Cursors, Newline display character}**.

VEDIT does not automatically reformat paragraphs when you change the right margin, e.g. resize the window. You can reformat paragraphs with **{EDIT, Format paragraph}** (**<Ctrl-B>** or toolbar).

Instead of using horizontal scrolling, VEDIT can also wrap long lines onto multiple screen lines. This is independent of word wrap. See the topic "Wrapping Long Lines on the Screen" in Chapter 4.

Another popular way to handle long text lines is to wrap them onto multiple lines at word boundaries. This option can be selected with **{VIEW, Word wrap display}** (or toolbar).

Deleting Text

As is typical of most editors, pressing **<Del>** deletes the character at the cursor; pressing **<Backspace>** deletes the character just *before* the cursor.

When the cursor is at the end of a line, pressing **<Del>** deletes the "newline" character, effectively appending the following line.

Similarly, when the cursor is at the beginning of a line, pressing **<Backspace>** deletes the previous "newline". This appends the current line to the end of the previous line.

In short, when you're at the beginning of a line and want to append it to the previous line, press **<Backspace>**. When you are at the end of a line and want to append the following line, press **<Del>**.

(The behavior with respect to deleting "newlines" is configurable.)

Deleting Lines

VEDIT has three functions for deleting partial or entire lines:

<Ctrl-J>	**[ERASE BOL]**	Erase to beginning-of-line
<Ctrl-K>	**[ERASE EOL]**	Erase to end-of-line
<Ctrl-L>	**[ERASE LINE]**	Erase (delete) entire line

Press **<Ctrl-L>** to delete an entire line; any following lines will then move up.

Press **<Ctrl-K>** to delete text from the cursor position to the end of the line. You can "blank out" the current line by first pressing **<Home>** and then **<Ctrl-K>**. This is different from **<Ctrl-L>** in that it changes a line of text to a blank (empty) line instead of erasing it entirely.

Deleting Words

VEDIT can delete the word before or after the cursor position.

<Ctrl-Bksp>	**[DEL PREV WORD]**	Delete previous word.
<Ctrl-Del>	**[DEL NEXT WORD]**	Delete next word.

Pressed once, it deletes the characters that make up the word; pressed again, it deletes the space(s) between the words. Therefore, you must press these functions twice for each word to be deleted.

Moving the Cursor

VEDIT has a full range of cursor movement functions in addition to the four basic "arrow" keys. You can move the cursor by words, paragraphs, pages and to the beginning or end of a line or screen. Other movements using the **{GOTO}** menu are described in Chapter 6.

Moving past the beginning/end of a line

Pressing **<Cursor Left>** at the beginning of a line, moves the cursor to the end of the previous line. Similarly, pressing **<Cursor Right>** at the end of a line, moves to the beginning of the following line. (This behavior is configurable.)

You can configure how VEDIT handles the cursor when it is past the end of a line, e.g. what happens when you press **<Cursor Down>** from a long line past a short line. By default, VEDIT allows the cursor past the end of a short line, but as soon as you type, it zips the cursor left to the "real" text. Alternatively, VEDIT can pad the short line with enough spaces to reach the cursor position.

See **{CONFIG, Emulation, Cursor position mode}** in Chapter 6 for a more detailed description.

Moving by Words

You can move forwards and backwards by words:

<Ctrl-Right Arrow>	**[NEXT WORD]**
<Ctrl-Left Arrow>	**[PREV 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 a line, **<End>** moves the cursor 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 a line, **<Home>** moves the cursor to the beginning of the previous line.

NOTE: Once they are used to it, most users like having **<End>** move the cursor to the ends of successive lines. If you don't, you can change this behavior with **{CONFIG, Emulation, [LINE BEGIN/END] emulation mode}**.

Moving by Pages

Well, actually just less than a screen "page". VEDIT shows some overlap between "pages" for easier reading. This overlap is about two lines in a typical 20 line window. The overlap is larger for bigger windows; there is no overlap for very small windows.

<PgDn> **[PAGE DOWN]**

<PgUp> **[PAGE UP]**

Even though these functions refer to "pages", they bear no relation to printed pages of a document. They are simply "screen" pages and their size depends on the number of lines in the current window.

Moving by Paragraphs

What is a paragraph? Unlike most word processors, VEDIT does not use unseen control characters to mark where one paragraph ends and the next begins. Instead, it simply considers a blank line as a break between paragraphs. (See "Word Processing" in Chapter 4 for some exceptions.)

<Ctrl-Down Arrow> **[NEXT PARAGRAPH]**

<Ctrl-Up Arrow> **[PREV PARAGRAPH]**

[PREV PARAGRAPH] first moves the cursor to the beginning of the current paragraph. Pressed again, it moves to the beginning of the previous paragraph. **[NEXT PARAGRAPH]** also moves the cursor to the first character of the next paragraph.

Undo and Redo

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 have VEDIT "undo" those changes.

To undo your last action press:

<Ctrl-Z> or <Alt-Backspace>

Each press of **<Ctrl-Z>** undoes the next previous action. Actions that 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 **<Ctrl-Z>**.

To try it, first delete a line of text by pressing **<Ctrl-L>**. Then press **<Ctrl-Z>** and the line comes back. Go ahead and make some other changes — deleting, overwriting and inserting text. Repeatedly pressing **<Ctrl-Z>** will reverse these changes step by step.

To undo all changes to the current line press:

<Ctrl-Shift-Z>

Pressing **<Ctrl-Shift-Z>** again just moves the cursor to the last line you were on. Pressing it again then undoes the changes to that line, and so on. This is a quick way to undo the changes you just made line-by-line. Pressing **<Ctrl-Shift-Z>** is equivalent to pressing **<Ctrl-Z>** several (or many) times.

Sometimes when you use Undo you accidentally go too far and remove changes you wanted to keep. Or you simply change your mind and want to "undo" the undo. This capability is called "Redo".

To redo your last undo press:

<Ctrl-Y>

To try it out, enter a few short lines of new text. Then repeatedly press **<Ctrl-Shift-Z>** (the undo line function) until the new text is gone. Now repeatedly press (or hold down) **<Ctrl-Y>**; your text should come back character by character.

Since each **<Ctrl-Y>** will redo one **<Ctrl-Z>**, it usually takes several (or many) **<Ctrl-Y>**'s to redo one **<Ctrl-Shift-Z>**.

Repeating Operations

It is often desirable to repeat an editing operation such as inserting the same character many times. With the **[REPEAT]** function (<**Ctrl-R**>), you can perform these operations a specified number of times without having to press the same key over and over again.

For example, if you needed to enter 50 "*" (asterisk) characters into your text, you could press the "*" key fifty times, counting very carefully. However, the better way is to use the **[REPEAT]** function.

➢ **Example - To enter 50 "*" characters into your text:**

1. Press <**Ctrl-R**>. The status line will prompt you with:

   `Enter repeat count:`

2. Type "50" and press <**Enter**>. The status line now prompts:

   `Press key to repeat:`

3. Press "*". Fifty "*" (asterisk) characters will be entered into your text.

At the "Enter repeat count" you can enter any number up to 256. (This maximum of 256 is configurable up to 65,535.)

After the repeat count is entered, you can press any text character or press any editing function such as **[ERASE LINE]**.

For example, to delete 75 lines, press <**Ctrl-R**>. At the prompt type "75" and <**Enter**>. Then press <**Ctrl-L**> (the **[ERASE LINE]** function).

You can also repeat the last editing operation by pressing **[REPEAT LAST]** (<**Alt-R**>). Each successive **[REPEAT LAST]** repeats the last editing operation one more time.

VEDIT also lets you repeatedly perform a sequence of editing steps over and over again. This is done using "*keystroke macros*" which are extensively covered in their own topic in Chapter 4 (Editing Guide).

"Cut and Paste" a Block

VEDIT has an exceptionally wide range of block operations. You can see most of them in the **{BLOCK}** and **{BLOCK, Edit/translate}** menus.

Probably the most common block operation is to "cut" a block from one location and "paste" it into another location. Most Windows editors give you just one way. VEDIT gives you at least three ways:

- You can "cut and paste" using the Windows "*clipboard*". This is identical to other editors.
- You can "cut and paste" using any of VEDIT's 100 "*text registers*". For simplicity, one text register is reserved as the "*scratchpad*".
- You can directly copy or move a highlighted block to the current cursor position.

We suggest using VEDIT's "*scratchpad*" or other text registers instead of the Windows "*clipboard*" for copying blocks within VEDIT.

Marking (highlighting) a Block

VEDIT gives you three main ways to mark a block of text:

- Use the mouse to "drag" over the selected text.
- Hold down the **<Shift>** key while moving the cursor.
- Select **{BLOCK, Set stream marker}** in the menus, or press the equivalent hot-key **<F9>**, or select it from the toolbar.

Since you probably already know how to highlight a block with the mouse or **<Shift>** key, let's highlight a block using the hot-key for **{BLOCK, Set stream marker}**.

➢ **To highlight a block using a "hot-key":**

1. Position the cursor on the first character to be included in the block.
2. Press **<F9>** to set the first block marker. Or select "Set stream marker" from the toolbar. Note the message "1-END" on the status line.
3. Move the cursor to the end of the block; any desired method can be used. For example, you could search for text at the end of the block.

   Position the cursor just past the last character of the block. To include the "newline" at the end of a line, position the cursor at the beginning of the next line.
4. Press **<F9>** again to set the second block marker. Note the message "BLOCK" on the status line.

For most block operations, you do not have to explicitly set the second block marker. The cursor position is used as the second block marker.

After marking a block of text, you will notice that VEDIT's blocks are "persistent". A block remains marked until you process it or explicitly remove the markers. One advantage is that you can directly copy the block to another location using **{BLOCK, Copy to cursor}**.

If you decide not to use a marked block, you have to explicitly remove the markers. VEDIT makes this as easy as possible:

- Simultaneously press both mouse buttons.
- Double-press the **<Ctrl>** key.
- Press **[CANCEL]** (**<Ctrl-\>**).
- Select **{BLOCK, Remove markers}** (**<Shift-F9>** or toolbar) or **{ESCAPE, Remove block markers}**.

Directly copying a block

After marking a block, you can directly copy or move it to the current cursor position without using the clipboard or a text register.

➢ **To directly copy/move a block.**

1. Highlight the block to be copied. Use any method desired. For example, follow steps 1. through 4. above.
2. Move the cursor to the destination for the block.
3. Press <**Ctrl-F9**> to copy the block to the cursor position.

   Alternatively, press <**Alt-F9**> to move the block.

"Cut and Paste" with the VEDIT "Scratchpad"

Since you probably already know how to "cut and paste" using the Window's clipboard, let's use the VEDIT "scratchpad" instead.

➢ **To "cut and paste" with the "scratchpad":**

1. Highlight the block to be "cut". Use any method desired. For example, follow steps 1. through 4. on the previous page.
2. Press <**Numpad-**> or <**Alt-F11**>, the equivalent hot-keys for **{BLOCK, Move to register}**.

   At the "Move to register" dialog box, simply press <**Numpad-**> again to select the default register "0", also called the "*scratchpad*".

   Once you get used to it, you simply double-press <**Numpad-**>.

   Alternatively, press <**Numpad+**> to copy the block.
3. Move the cursor to the destination for the block.
4. Similarly, double-press <**Numpad***> or <**F11**>, the hot-keys for **{BLOCK, Insert register}**.

HINT: If you double-press **<Numpad+>** or **<Numpad->** when no block is highlighted, it copies/moves the current line to the scratchpad. Then move the cursor and double-press **<Numpad*>** to insert the line. This makes it very easy to copy or move a single line.

Notes:

You may have noticed that there is also an **{EDIT, Scratchpad}** sub-menu and scratchpad icons on the toolbar. We wanted to demonstrate the text register selection dialog box and how easy it is to select the scratchpad.

As you become more familiar with VEDIT, you will probably use additional text registers. For example, registers "1" through "9" can be used to store small blocks of text that are inserted over and over again.

Printing Text

You can print the entire file or just a highlighted block.

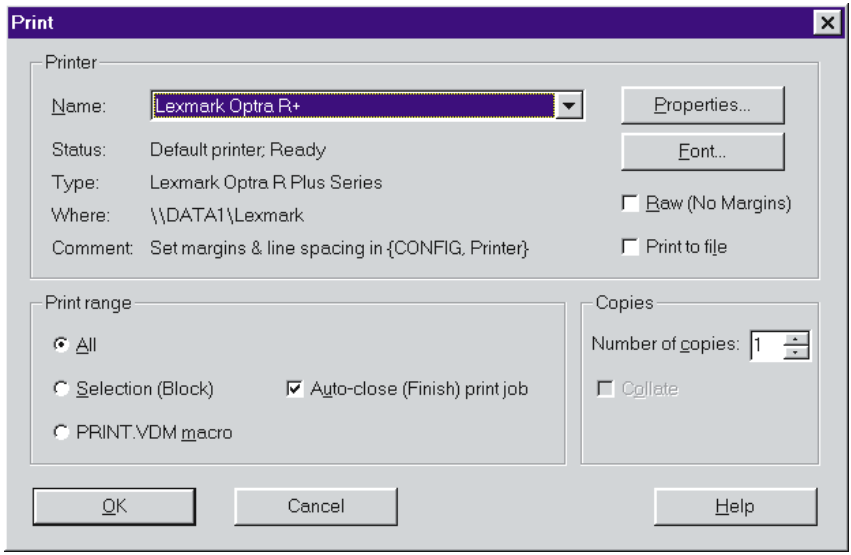
➢ **To print the entire file:**

1. Select **{FILE, Print}** (default: **<Ctrl-P>**).
2. Assuming a block of text is not highlighted, "**All**" is automatically selected; otherwise manually select "**All**".

   If you prefer to have the filename, page number and date printed at the top of each page, select "**PRINT.VDM macro**" instead.
3. If needed, select **[Setup]** to select a different printer or change its properties.

   If desired, select **[Font]** to select the font used to print the entire file (or block).
4. Select the **[Ok]** button or press **<Enter>**. The entire file should print.

You can change VEDIT's top, bottom and left margins for printing with **{CONFIG, Printer}**. You can also select single, double or triple spacing.



Notes:

The **[Font]** button lets you print the text in any desired font and size.

The "**[ ] Raw**" option lets you print a file without margins or other processing by VEDIT.

Disabling the "**[ ] Auto-close print job**" option lets you print two or more blocks on the same page.

The topic "Printing" in Chapter 4 describes printing in more detail.

Saving Your Work

You should periodically save your work to disk. This protects you against power failures, system "crashes" and mistakes on your part.

➢ **To save your work and continue editing:**

1. Select **{FILE, Save}** (default: **<Alt-F12>**).

-OR-

1. Select **{FILE, Save all}** if you are editing multiple files and want to save all of them,.

Notes:

You can also let VEDIT automatically save your work, say every 20 minutes by setting **{CONFIG, File handling, Auto-save interval}**. This is described under "Auto-file Save" in Chapter 4.

Exiting VEDIT

➢ **To save your work and exit VEDIT:**

1. Select **{FILE, Exit}** (**<Alt-F4>** or **<Alt-X>**).

-OR-

1. Press the **<Esc>** key to bring up the **{ESCAPE}** menu and select "Exit (save/abandon)".
2. If any modified files are open in VEDIT, you will be prompted whether to save or abandon them. A typical prompt would be:

   `Save NEWS.TXT? [Yes] [No] [Save-all] [Quit-all]`

   Select **[Yes]** to save this file

   Select **[No]** to abandon (quit) this file without saving changes.

   Select **[Save-all]** to save all remaining files and exit VEDIT.

   Select **[Quit-all]** to abandon all remaining files and exit VEDIT.

Notes:

See the topic "Exiting" in Chapter 4 for more information.

That's It!

That's all you need to create and edit text files. Obviously there is much more VEDIT can do for you. Take some time to get comfortable with basic editing, then read the more detailed chapters that follow.

We have enjoyed creating VEDIT and sincerely hope it serves you well.

Chapter 4

Editing Guide

Starting (Invoking) VEDIT

The details of starting VEDIT are different for the Windows and non-Windows (DOS, UNIX, QNX) versions. However, several aspects are the same for all:

- VEDIT can be started without an initial filename. All files are then opened with **{FILE, Open}**.
- VEDIT can also be started with one or more initial filenames. Each of the files is opened for immediate editing.
- Various invocation options can be selected. For example, you can start VEDIT in read-only mode, skip loading the configuration files, or auto-execute a macro such as WILDFILE.VDM.

Starting VEDIT for Windows

There are many ways to start VEDIT under Windows.

- Like all Windows programs, you can start VEDIT by double-clicking its icon with the mouse. If **{FILE, Enable edit restore}** was enabled, this starts VEDIT with the previously loaded files; otherwise, it start VEDIT without any open files.
- Using *"drag and drop"*, drag the desired file(s) to the VEDIT icon.

  In Windows, you can also right-click on a file and select "Copy", then right-click on the VEDIT icon and select "Paste". This is an alternative to "drag and drop" when the file icons and the VEDIT icon are not visible at the same time.
- In Windows, you can double-click on files whose "type" or filename extension is associated with VEDIT.
- Select the "Run" command and enter the name of the executable VEDIT, any desired invocation options and any desired filenames. This is one way to start up VEDIT with invocation options. (You typically must specify the full pathname to the executable VEDIT.)
- In Windows 95/98/NT, you can start VEDIT from the MS-DOS command prompt "C:" in the same way as with the "Run" command above. You can start either the Windows or the DOS version.

Changing the VEDIT Icon Properties

The VEDIT icon properties can be changed to select invocation options. Typically, you would copy the main VEDIT icon and then change the new icon's properties.

The following example demonstrates how to create a new VEDIT icon which uses the "**-e**" invocation option to ignore any previously saved "edit session restore", i.e. if you just double-click the icon, VEDIT will start without any open files. (Otherwise it may start with the last files you were editing.)

The icon properties can also set the default directory for the first File-open dialog box, in this example "c:\projects\webs\cafe".

➢ **To create a custom VEDIT icon:**

1. Select the VEDIT program group (or folder) which contains the main VEDIT icon.

   While holding down the **<Ctrl>** key, make a copy of the VEDIT icon by dragging it to a new location within the same group.

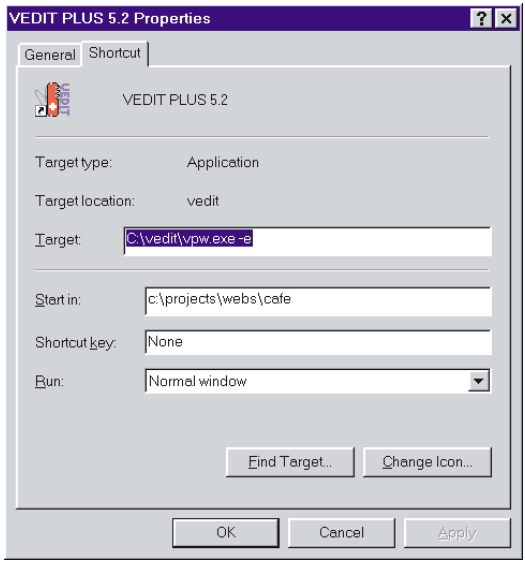
2. Open the icon's properties dialog box. In Windows 95/98/NT, right-click the icon and select "Properties".

3. Change the "Target" to include any desired invocation options, in this example "**-e**".

4. Change the "Start in" directory to the default directory that the first File-open dialog box will start in.

5. If desired, select "Change icon" to pick a different icon.

6. In Windows 95/98/NT, close the dialog box. If desired, right-click the icon, select "Rename" and give it a new name.



Starting VEDIT for DOS

Begin editing by typing the name of the VEDIT program; this is normally "**vedit**". Although not necessary, this is normally followed by the name of the file or files you want to edit or create. For example:

vedit letter.txt

vedit chapter1.txt chapter2.txt

vedit *.c *.h

When creating a new file, the message "New file" is temporarily displayed on the status line.

To see a summary of all DOS version invocation options, give the command:

vedit /?

Invocation Options (All Versions)

VEDIT supports numerous invocation options. The syntax is:

vpw +*options* -*options filename* -a *outfile*

(VEDIT PLUS for Windows 3.1 is "**veditpw**". VEDIT PLUS for DOS is "**vedit**".)

In the Windows version, invocation options can be specified by changing the icon's properties or by using the "Run" command.

In addition to the filename(s), one or more *invocation options* can be specified (in upper or lower case) when you start up VEDIT:

-b Puts VEDIT into "*browse-only*" mode. All files are opened in Read-only mode; you can view them, but cannot alter them.

-c *command* Executes the VEDIT PLUS macro language commands '*command*'. The '*command*' may be delimited with quotation marks ("); otherwise it ends on the first space. The commands are executed ***after*** the **`startup.vdm`** file.

+c *command* Same as "**-c**", except that the commands are executed ***before*** the **`startup.vdm`** file.

-d Disables Windows 95/98/NT long filename support. VEDIT will only recognize the short 8.3 names.

-e Disables the edit session restore feature. Only needed when invoking VEDIT without any filenames and when VEDIT was last exited with **{FILE, Enable edit restore}** enabled.

-g Disables auto-configuration; the **`vedit.cfg`** and **`vedit.key`** files are not loaded during startup.

-g *n* (DOS version only) Overrides the value of **{CONFIG, Misc, Auto-load config}** built into VEDIT with the value of '*n*'. "-g3" therefore forces auto-configuration.

-i *execfile* '*execfile*' is executed in place of the **`startup.vdm`** file. The option "**-i xxx**", where **xxx** is a non-existent file, can be used to start VEDIT without any startup file.

-j (DOS version only) Enables VEDIT's normal keyboard polling. Same as setting **{CONFIG, Misc, Keyboard polling}** to "0" for maximum compatibility with other DOS programs (TSRs). Only needed if the keyboard polling has been set to "1" and VEDIT crashes on startup.

-k (Windows version only) Disables reading the **`vedit.ini`** file on startup and saving a new **`vedit.ini`** file when exiting. A default size and font are used on startup; the final size and font are not saved.

(DOS version only) Disables the keyboard repeat (typematic) rate speedup. Same as setting **{CONFIG, Misc, Keyboard repeat rate}** to "0". This setting is needed if VEDIT appears slow to startup, slow to exit, or appears to "lockup" the computer on exit. See "Setting the Keyboard Typematic Rate" in Chapter 2 (Getting Started).

-k*inifile* '*inifile*' is used as the Windows parameter file in place of the normal **`vedit.ini`** file. The full pathname should be specified.

NOTE: There **must not** be a space between the "-k" and the filename.

-m (DOS only) VEDIT uses the monochrome (Black/White) screen attributes. (You can configure both a set of color attributes and a set of monochrome attributes.) This is only needed if VEDIT was installed (configured) for a color display and you have a monochrome display connected to a color (VGA) adapter.

-n *nnn* Passes the numeric value '*nnn*' to a command macro as an option. For example, the SORT.VDM macro uses this option. (VEDIT PLUS: This value is accessed with the **N_Option** command.)

-o (DOS version only) VEDIT writes to the screen via the "BIOS" instead of directly to the screen memory. This overrides the value set by the configuration parameter **Config(H_USE_BIOS)** (Chapter 8). Same as "**-o1**".

-o2 VEDIT writes to the screen via the "BIOS" on non-IBM type machines. This is necessary for VEDIT to work on the Tandy 2000 and other "near" compatibles.

-p Prints the specified file and immediately exits. This is equivalent to selecting **{FILE, Print}** and then "All - entire file".

-q Runs VEDIT in "quiet" (minimized) mode without displaying it on the screen. It is typically followed by the "**-x**" or "**-c**"

options to execute a macro. When the macro is done, it automatically saves all files and exits.

-r Restricted: only the file(s) specified on startup can be edited. All editing operations can be performed, but additional files cannot be opened. Useful when shelling to VEDIT from another program where you don't want the user to be able to edit any files other than the specified ones.

-s *nnn* (DOS, QNX, UNIX only) Restricts VEDIT to use a maximum of '*nnn*' Kbytes of memory for all data areas. See the on-line help for details about this rarely used option.

-v Start up VEDIT PLUS at the "COMMAND:" prompt.

-x *execfile* '*execfile*' is loaded into text register 100 and executed as a VEDIT PLUS command macro. If no filename extension is given ".VDM" is assumed. '*execfile*' is executed ***after*** the **`startup.vdm`** file. This is the normal way to auto-execute command macros.

+x *execfile* Same as "**-x**", except that '*execfile*' is executed ***before*** the **`startup.vdm`** file.

— Signals the end of the options. This is only needed when the filename being edited begins with a "-".

filename The name of the file or files to edit. The wildcard characters "*" and "?" may be used to load a group of files. Long filenames containing spaces or commas must be enclosed in double-quotes. All filenames may include full drive and path specifications. Each single filename may be followed by the following two options:

-a *outfile* The preceding file being edited will be saved under the name '*outfile*'. This is similar to using the **{FILE, Save as}** function.

-l *nnn* Start editing the preceding file on line '*nnn*'. If there are fewer than '*nnn*' lines in the file, it will start at the end of the file. You can specify a numeric expression such as "-l(7890/3)".

NOTE: When opening just one file, you can also use the following syntax to open the file on line '*nnn*':

vpw -*nnn filename*

Notes:

All options beginning with "+" must be specified ***before*** any options beginning with "-". The **`startup.vdm`** file, if enabled and found, will be executed after any "+" options and before any "-" options.

DOS users who need as much memory as possible for running programs such as compilers from within VEDIT should install the supplied V-SWAP program into memory before starting up VEDIT.

See Also:

"Exiting VEDIT" and "Backup Files" later in this chapter.
"Auto-execution" in Chapter 5 (Advanced Topics).
{MISC, DOS shell}, in Chapter 6 (Menu Reference).

Examples:

To edit the file "CONFIG.SYS" in the root directory:

vpw c:\config.sys

After you have modified and saved CONFIG.SYS, you will have both an updated copy called CONFIG.SYS and the original copy called CONFIG.BAK.

To edit the file with the long filename "a long filename" use double-quotes:

vpw "a long filename"

To edit the file "PROGRAM.C" and have the cursor start up on line 984, type:

vpw program.c -l 984

-OR-

vpw -984 program.c

To load the existing file "FAXSHEET.SAV", make changes to it and save it as the file "TOMJONES.292":

vpw faxsheet.sav -a tomjones.292

VEDIT Environment Variable

If you use the same invocation options over and over again, you can set up the environment variable "VEDIT" with any desired default options.

Any options specified by this environment variable are processed by VEDIT before those given on the command line. Therefore, the options specified by the environment variable should usually be preceded with "+".

For example, if you always want VEDIT to use **`toms.vdm`** as the startup file instead of the default **`startup.vdm`**, you could add the following line to your AUTOEXEC.BAT file:

SET VEDIT=+I C:\VEDIT\TOMS.VDM

Loading Multiple Files

You can start up VEDIT with up to 32 files. The wildcards "*" and "?" can also be used.

The first file is loaded into the main edit buffer #1, the second into buffer #2, the third into buffer #3, etc. **{FILE, Previous buffer}** (**<F5>**) and **{FILE, Next buffer}** (**<F6>** or toolbar) will toggle between the files.

Examples:

To load the files "FILE1", "FILE2" and "FILE3", type:

vpw file1 file2 file3

To load the same files, but save "FILE2" as "NEWFILE2", type:

vpw file1 file2 -a newfile2 file3

The "**-a**" option may be used with multiple files. For example, to load the file "CHAP1.TXT" and save it as "CHAP1.NEW", and load "CHAP2.TXT" and save it as "CHAP2.NEW", type:

vpw chap1.txt -a chap1.new chap2.txt -a chap2.new

To load all files in the current directory with extensions ".C" and ".H", type:

vpw *.c *.h

You can also load multiple files from the **{FILE, Open}** dialog box. In the Windows 3.1, DOS and QNX versions, you can load multiple files using wildcards by selecting the "**[ ] Load multiple files**" box.

VEDIT PLUS: You can load multiple files from Command Mode with the **File_Open( )** command. For example: **File_Open("chapter*.txt")**.

Read-only (Browse-only) Mode

To view a file you don't want to accidently alter, enable the "**[x] Read-only**" option in the file open dialog box. This has the added benefit of letting you navigate through the file more quickly. **{GOTO, Beginning of file}**, **{GOTO, End of file}** and relative jumps using the mouse will then work instantly on even multi-megabyte files.

Files opened in read-only mode are indicated with a "!" preceding their filename on the status line.

CD-ROM and other "read-only" files are automatically opened in read-only mode and can be quickly navigated.

If VEDIT is invoked with the "**-b**" option, all files are opened in read-only mode and the "**[x] Read-only**" option is always enabled.

You can temporarily prevent making any changes to a file by selecting **{FILE, Browse mode}**. The current file then cannot be altered any further. However this is not quite the same as read-only mode, because navigating is not as fast; you can also disable browse mode when desired.

Notes:

You cannot disable browse mode if you invoked VEDIT with "**-b**" or opened the file in read-only mode.

VEDIT does not always know the current line number when navigating a read-only file; the line number is then displayed as "?????". If you must know the line number in a browsed file, set **{CONFIG, File Handling, Enable fast**

browse mode} to "No". Browsing will be slower, but the correct line number will always be displayed.

See Also:

{FILE, Browse mode} in Chapter 6 (Menu Reference).

Overwrite-Only Mode

When editing some types of files, particularly binary (e.g. .EXE) and database (e.g. .DBF), it is important not to change the size of the file or the file would become corrupted and unusable.

In "overwrite-only" mode, deletions and insertions that would change the file size are not allowed. Only character overstriking, block overwrites, and other editing operations that don't change the file size are allowed. Search/replace operations are allowed, but only when the replacement text is the same size as the search text.

Overwrite-only mode is controlled with **{CONFIG, File handling, Overwrite-only mode}**. It has three settings:

0. Disabled
1. (Default) Record mode. Overwrite-only mode is only enabled if the "File type" is set to 8 or greater for binary or database file editing.
2. Enabled for all file types.

The default value of "1" only enables overwrite-only mode when **{CONFIG, File handling, File type}** is set to "8" or greater. This is the normal setting for editing binary and data files, which are precisely the types of files whose size should not be changed.

Therefore, by default, VEDIT selects overwrite-only mode when editing binary/data files.

Notes:

In Overwrite-only mode, **{BLOCK, Copy to cursor}**, **{BLOCK, Move to cursor}** and **{BLOCK, Insert register}** always overwrite the existing text at the cursor.

Similarly, **{BLOCK, Move to cursor}** and **{BLOCK, Move to register}** fill the original source block with the configurable "block fill" character, typically spaces. **{EDIT, Delete}** also fills the block instead of deleting it.

If you disable **{CONFIG, Config all buffers}**, you can have overwrite-only mode enabled for some of the files you are editing and not for others.

(VEDIT PLUS DOS version only) Disk sector editing is always in overwrite-only mode.

See Also:

The topic "File Types" later in this chapter.

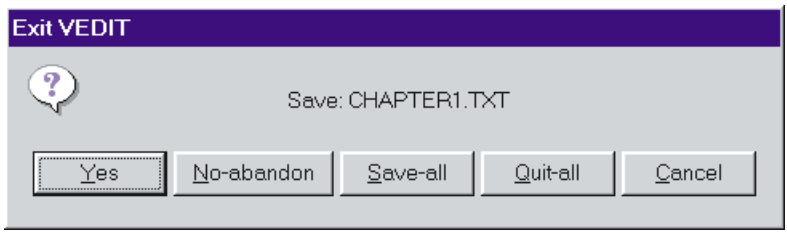
Exiting VEDIT

VEDIT gives you a great deal of flexibility when exiting, especially when you are editing multiple files. You can selectively save or abandon each file (buffer) that has been modified, or save or abandon all files at once.

➢ **To exit VEDIT and save or abandon the current file(s):**

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

   If no modified files are open in VEDIT, this will immediately exit VEDIT. Otherwise it displays each modified file and prompts whether it is to be saved or abandoned. It repeats this for each file and exits VEDIT. A typical prompt would be:



2. Select **[Yes]** to save this file. If the file has no assigned filename, you are prompted for one.

   Select **[No]** to abandon (quit) this file; the changes are not saved.

   Select **[Save-all]** to save all remaining files and exit VEDIT. (Note: it only saves those buffers that have assigned filenames!)

   Select **[Quit-all]** to abandon all remaining files and exit VEDIT. If there are additional modified files open, you are prompted for confirmation.

Notes:

If **{FILE, Enabled edit restore}** is set, VEDIT saves the names of the files being edited, so that the files can automatically be reloaded later. Files which you now abandon will be reloaded with their previous contents. Files which you now save will be reloaded with their current contents.

When you abandon a file, you only discard any changes made to the text since the last time you saved the file (e.g. with **{FILE, Save and continue}**) or the auto-save feature saved all files.

VEDIT checks to see if any changes have been made since the file was opened or last saved, and only prompts for modified files.

If there is insufficient disk space to save the modified file, VEDIT displays an error and aborts the request to exit. If this happens, you have several options. You can quit and abandon any changes made or delete unneeded files from your disk. Or you can use **{FILE, Save as}** to save the file on another drive.

{MISC, DOS shell} and **{MISC, Run program}** let you execute DOS commands for deleting files. Be careful not to delete any files that 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} in Chapter 6 (Menu Reference).

Edit Session Restore

When **{FILE, Enable edit restore}** is set, VEDIT saves its entire status when you exit. Subsequently invoking VEDIT without filenames resumes your previous edit session, just as if you had never exited.

However, if you invoke VEDIT with filenames, the previous edit session status is not used. To invoke VEDIT without filenames and without restoring the previous edit session, use the "**-e**" option, e.g. "**vpw -e**".

VEDIT's status is saved in the files **`veditsav.env`** and **`veditsav.dat`**. Depending upon the setting of **{CONFIG, File handling, Save session in current directory}**, these two files are either stored in the current directory or in the *User Config Directory*, typically **`c:\vedit`**.

When the edit session is saved in the *User Config Directory*, only one (the last) session can be saved. If you then invoke VEDIT (without filenames) from anywhere, you will be switched to the last directory you were in and the files you were last editing. This is the default setting.

When the edit session is saved in the current directory, you can save multiple sessions, each in a different directory. If you then invoke VEDIT without filenames, it will restore the last edit session you had in that directory.

When VEDIT starts up, it searches for the files **`veditsav.env`** and **`veditsav.dat`** first in the current directory and then in the *User Config Directory* regardless of how **{CONFIG, File handling, Save session in current directory}** is set. Once the files are found, they are loaded and then immediately erased.

Notes:

Files that you abandon during **{FILE, Exit}** will be reloaded with their previous contents. Files which you save will be reloaded with their current contents.

Your entire configuration is saved/restored including changes that you did not make permanent with **{CONFIG, Save into VEDIT}**. This includes any changes to the keyboard layout. (Exception: The setting of **{FILE, Enable edit restore}** is not saved/restored.)

The only status not saved is the history of previously entered filenames, search strings, i.e. the character strings you can recall with **[CURSOR UP]**.

The status save is performed by the macro **`veditsav.vdm`**.

Backup Files

When you edit and modify an existing file, VEDIT can optionally create a backup of the original file in one of two ways:

- (Default) Rename the original file to have a ".BAK" filename extension. Any existing ".BAK" file is deleted during this process.
- Move the original file to the *VEDIT Backup Directory*, typically **`c:\vedit\backup`** or **`c:\backup`**. Any existing backup file by the same name is deleted during this process.

Backup files take up additional disk space but provide important data protection. They provide a backup in case you make a major editing mistake that you may not notice until days later. They also provide some protection against the accidental deletion of files — the most recent revisions may be lost, but at least the previous revisions are still there.

Which backup method you use is a matter of personal preference. Each method has its advantages and disadvantages:

Rename Method Advantages:

- If you edit two files with exactly the same name in two directories, you will have a backup of each.
- You can easily view the backup files in the current directory.
- Always fastest.

Rename Method Disadvantages:

- If you edit files in one directory with the same name, but different extensions, e.g. "prog.c", "prog.inc" and "prog.h", you will only have a backup of one of them, and you cannot predict which one.
- You end up with .BAK backup files all over the disk.

Move Method Advantages:

- If you edit files in one directory with the same name, but different extensions, e.g. "prog.c", "prog.inc" and "prog.h", you will have a backup of each one.
- All backup files are in one directory where they can easily be deleted.

Move Method Disadvantages:

- If you edit two files with exactly the same name in two directories, you will only have a backup of the last one saved.
- When editing files on other drives, file saving is slower because the original file must be copied to the Backup directory. (The move is instantaneous on the same drive.)

If desired, you can delete the backup files when you are sure that you do not need them anymore. With the move method, you can simply delete all files in the Backup directory. With the rename method, you must delete all "*.BAK" files in each directory.

➢ **To choose the backup method (or turn backups off):**

1. Select **{CONFIG, File handling, Backup-file mode}**. Enter the value:

   0 - turn off backup files entirely
   1 - create backups by renaming original file with ".BAK" extension
   2 - create backups by moving original file to the VEDIT Backup directory

2. Skip this step if you want to use the default *VEDIT Backup Directory* of "c:\vedit\backup". However, you may prefer to use "c:\backup" or another directory.

   (DOS version) Set the desired *VEDIT Backup Directory* with **{CONFIG, Misc, Change VEDIT Backup directory}**. To make these changes permanent in VEDIT, select **{CONFIG, Save config}**. It is a good idea to also select **{CONFIG, Misc, Save into VEDIT.EXE}**.

   (Windows version) Set the desired *VEDIT Backup Directory* by editing the file **`vedit.ini`**. Change the item "BackupDir" to the desired directory.

   Be sure to set the drive too; otherwise you will receive an error when a directory, such as "\vedit\backup", does not exist on the current drive.

DOS: You can override the configured VEDIT Backup directory with the environment variable "VBACKUP".

Auto-file Save

VEDIT can optionally auto-save all modified files after a configurable number of minutes. By default, this feature is turned off.

➢ **To enable auto-save:**

1. Select **{CONFIG, File handling, Auto-save interval}** and enter the desired number of minutes. A typical value is "20".

2. To ensure that the configuration change is permanent, select **{CONFIG, Save config}**.

VEDIT will auto-save after the configured number of minutes from the time of the last auto-save ***-or-*** you manually selected **{FILE, Save all}**.

We highly recommend that you enable auto-save set to around 20 minutes! Just be careful not to accidentally alter files you don't want to alter, or VEDIT may auto-save the undesired alteration. You should open files you don't want to alter in "read-only" mode. (See "Read-Only Mode" earlier in this chapter.)

The Status Line

VEDIT provides a wide variety of information on the "Status Line". The left side of the status line is used as a message area. When there is no message to display, the current key assignment to the **[HELP]** function is displayed.

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 number of the active edit buffer.

BLOCK Indicates that both block markers have been set. The block markers can be removed by pressing **[CANCEL]** (**<Ctrl-\>**) or **<Ctrl-Break>** or by selecting **{BLOCK, Remove markers}** (**<Shft-F9>**). They can also be removed via the toolbar, by pressing both mouse buttons or just double-pressing the **<Ctrl>** key.

BYTE: Indicates which byte from the beginning of the line/record is being editing. "COL:" changes to "BYTE:" when **{CONFIG, File handling, File type}** has been set to "8" or greater for editing binary and data files, or when editing in hexadecimal or octal mode.

C-N-S The status of the Caps/Num/Scroll Lock is displayed on the status line with the single letters "C", "N" and "S". (Windows and DOS only)

COL: *xxxx* Indicates in which column of the text the cursor is. Changes to "BYTE:" during record mode or hex/octal editing.

Command Mode Indicates that VEDIT is running a "command macro" (a macro written in the VEDIT PLUS macro language). Only the edit buffer number and filename 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 the topic "Exiting VEDIT" earlier in this chapter for more information.

EBCDIC The current window's display mode is set to EBCDIC. ASCII text will display as gibberish.

Note: If another translation table has been loaded, its name will be displayed in place of "EBCDIC".

filename Indicates the name of the file you are currently editing. The full pathname is displayed on the window's title bar. If the filename is extremely long, use **{HELP, Status display}** to display the full pathname.

	The filename is preceded with "*" when the file has been altered since the last time it was saved to disk.
	The filename is preceded with "!" when the file was opened in read-only mode. The file cannot be altered, but you can browse through it more quickly.
INS	Indicates that you are in "Insert" mode. Any typed text will be inserted and not overwrite the existing text. Otherwise, you are in "Overstrike" mode.
LINE: ***xxxx***	Indicates on what line of the file the cursor is. When the message is all in capital letters the entire file is in 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.
POS:*yyyy:xxxx*	Displays the cursor's offset into the file when editing in hexadecimal.
RM: ***xx***	Right margin value — only displayed if the right margin is set with **{CONFIG, Word processing, Right margin}**.
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 as described above for the "BLOCK" message.
<<	Indicates that the window is horizontally scrolled. The following number indicates how far it is scrolled.

User Interface

The Windows version works in the usual "Windows" manner with regard to menus, scroll bars, dialog boxes, mouse support, the toolbar and on-line help. The "normal" keyboard layout is very similar to most Microsoft and other products. However, the keyboard layout is completely configurable.

HINTS: Right click the mouse to pop-up a menu of common functions.

If the toolbar is not displayed, select **{VIEW, Toolbar}**. To display the toolbar on startup, select **{CONFIG, Display options, Enable toolbar}**.

Selecting Display Fonts (Windows version)

The display font used in the editing windows can be changed with **{View, Font}**. This displays the standard font selection dialog box; however, only "fixed width" fonts are listed and supported by VEDIT.

The fonts "VEDIT Oem", "VEDIT Ansi", "Fixedsys" and "Terminal" look and work best with VEDIT; each comes in several sizes. The True-Type font "Courier New" can be set to any desired size, but is of lower quality and slows down screen updates.

Fonts either display the "OEM" character set, in which character values 176 - 223 display the IBM PC line-drawing graphics, or the "ANSI" character set, in which character values 160 - 255 display non-english and special characters. The fonts "VEDIT Oem" and "Terminal" use the OEM character set. "VEDIT Ansi", "Fixedsys", "Courier" and most other fonts use the ANSI character set.

The custom "VEDIT Oem" font is the default. Instead of displaying control characters with the standard IBM PC "smiley face" and arrows, it displays more useful "^A", "CR", "LF", etc. Also values 0 and 255 are displayed as "Nul" and "FF Hex", instead of as spaces.

It is best to experiment to find a font you like; then select **{MISC, ASCII table}** to see how all characters are displayed. The font selection is saved into the **`vedit.ini`** file for the next time you run VEDIT.

By editing the **`vedit.ini`** file directly, you can display extra space between lines. OEM fonts tend to be tight; ANSI fonts tend to be looser. Change the "LineSpace" parameter from "0" to "1", "2" or more.

Shortcuts and Suggestions

VEDIT includes some unique shortcuts that save a few keystrokes and therefore speed your editing.

- As usual, you can mark a block of text with the mouse or with the **<Shift>** key. However, large blocks are more easily marked with **{BLOCK, Set stream marker}** (**<F9>** or the toolbar).

- Only use the Windows clipboard to transfer text between VEDIT and other programs. Use the VEDIT "scratchpad" and other text registers for cut and paste operations within VEDIT.
- The default text register is "0", also called the "scratchpad". In text register selection dialog boxes, you can immediately press **<Enter>** or simply double-press the function's hot-key to select the scratchpad.
- When no block is highlighted, **{BLOCK, Copy to register} (<Numpad+>** or the toolbar) copies the current line. Therefore, double-pressing **<Numpad+>** copies the current line to the scratchpad. Similarly, double-pressing **<Numpad*>** inserts the scratchpad at the cursor position. This is a fast way to copy an entire line.
- In the "File Open" dialog box, you can open several files at once by entering the filenames one after another, separated by commas. You can open a file on any desired line number with the "**-L**" option.
- At most prompts for a "number", e.g. **{GOTO, Line number}**, you can enter numeric expressions, such as **(12345+4589)/13** and hexadecimal numbers, such as **0xF47B9**.

  Variables and "internal values" from the VEDIT PLUS macro language can also be used. For example, to go to the exact middle of a file, select **{GOTO, File position}** and enter "**File_Size/2**".

DOS, UNIX and QNX Versions

The DOS, UNIX and QNX versions work very similar to the Windows version. The primary exceptions are that the on-line help works differently and there is no mouse Right-Click menu.

These versions have some special "shortcuts" not available in the Windows version. They are primarily designed to save keystrokes while navigating the menus and selecting default values in the dialog boxes.

- The on-line help topic "MENUS" describes the menu operation in detail. VEDIT has several special menu features including "sub-menu preview".
- The on-line help topic "DIALOG" describes the operation of the dialog boxes in detail. "Terse" or "Full" dialog boxes can be selected. Each dialog box also has its own on-line help.
- The on-line help topic "MOUSE" describes the DOS and QNX version's extensive mouse support. The right mouse button implements unique features such as variable-speed scrolling, quick jumps within the file, and "stealth" scroll bars.

Scrolling the Screen

Vertical Scrolling

The screen scrolls automatically as the cursor is moved towards the very top or bottom of the current window. By default, VEDIT scrolls when the cursor reaches about the third line from the top or bottom of a 24-line window. This ensures that you always see a few lines before and after the line you are editing. (VEDIT automatically adjusts this value according to the size of the current window.)

You can also use **[SCROLL UP]** (<**Alt-Up Arrow**>) and **[SCROLL DOWN]** (<**Alt-Down Arrow**>) to scroll the screen without having to move the cursor. This lets you view lines that are just off the screen.

Configuration Options:

The configuration parameters **Config(S_PG_OVERLAP)**, **Config (S_TOP_MARG)** and **Config(S_BOT_MARG)** determine how many lines of overlap you will see with **[PAGE UP]** and **[PAGE DOWN]**, and how close to the top/bottom of the window the cursor can get before the window scrolls. These parameters can only be changed by editing the **`vedit.cfg`** file as described in Chapter 8 (Configuration).

{CONFIG, Emulation, Special emulation modes} can change **[SCROLL UP]** and **[SCROLL DOWN]** to leave the cursor in the current screen line instead of in the current text line.

Horizontal Scrolling

VEDIT can deal with long lines in two different ways:

- (Default) Long lines can be viewed with horizontal scrolling. Each displayed line corresponds to one text line (or record).
- Long lines can be wrapped onto multiple window lines. An entire long line can be viewed at one time.

Horizontal scrolling is typically used for editing documents that 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 with **[SCROLL RIGHT]** (<**Alt-Cursor Right**>) and **[SCROLL LEFT]** (<**Alt-Cursor Left**>).

When scrolling sideways, the screen normally jumps 20 columns at a time. If desired, this can be changed with **{CONFIG, Display options, 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, Display options, Horizontal scroll margin}**.
2. At the prompt enter a new value of "1". This uses the current width of the window as the horizontal scroll margin.

   Alternatively, you can enter a value, such as 80 or 132, to only wrap lines longer than this value.

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 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 horizontal scroll margin can be set to values up to 2048 which is obviously wider than your screen. In this case, only lines longer than the scroll margin are wrapped, and normal horizontal scrolling is used to view columns up to the scroll margin. Since continuation lines stand out on the screen, this lets you flag lines longer than a specified length, e.g. lines that are too long for your application.

HINT: 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.

Configuration Options:

The continuation character can be changed with the configuration parameter **Config(S_CONT_C)** (see Chapter 8).

Notes:

Due to the continuation character, continuation lines display one character per line less than the first line.

UNIX and QNX versions: To reduce the amount of unwanted side to side scrolling, especially on slow CRT terminals, **[LINE END]** can be configured with **{CONFIG, Emulation, Line emulation mode}** to only go to the end of the window line instead of the end of the text line.

Screen Display & Keyboard Characters

VEDIT lets you display control and graphics characters in several different ways. You can also display and edit any file in hexadecimal (or octal). This is particularly useful for editing binary files.

EBCDIC (IBM mainframe) files can also be directly edited; they do not have to be translated to/from ASCII.

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

A computer's basic memory unit, called a "byte", allows 256 possible values for each character. 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 6 (Menu Reference).

In Windows and DOS, control and graphics characters can be entered directly by holding down the <**Alt**> key, typing the decimal value of the desired character on the keypad, and releasing the <**Alt**> key. All character values except "00" (the "Null" character) can be entered in this way. Control characters entered this way do not need to be preceded with **[ENTER CTRL]**.

Alternatively, you can enter control and graphics characters into your file using **{MISC, ASCII table}**. The "Null" character can be entered this way.

WINDOWS: To enter graphics (non-english) characters using the **<Alt>** key, you must precede the true decimal value with "**0**", to prevent Windows from translating the character from the IBM-PC (OEM) to the ANSI character sets; e.g. to enter the character with value 198, type **<Alt> 0198**.

DOS: You can enter the "Null" character (value 00) by pressing **[ENTER CTRL]** (**<Ctrl-Q>**) and then **<Ctrl-Shift-2>**.

Control and Graphics Character Display

Most control characters are normally displayed literally (i.e. the "smiling face" on an IBM PC). Alternatively, they can be displayed as a "^" followed by the corresponding letter, i.e. **<Ctrl-G>** displays as "^G".

Graphics characters are normally displayed literally, but can alternatively be displayed as a decimal value in the format <*nnn*>.

<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 by setting the configuration parameter **{CONFIG, Characters/Cursors, Tab display character}**.

<Ctrl-J> the Line-Feed character is the true "newline" character at the end of each line for Windows/DOS and UNIX style text files. The "newline" character is normally displayed as a space, effectively making it invisible. Alternatively, a visible "newline" character can be chosen with **{CONFIG, Characters/Cursors, Newline display character}**. With Windows/DOS text files, a Line-Feed not preceded by the normal Carriage-return character is displayed as "**<LF>**". With Mac text files, a Line-Feed is always displayed as "**<LF>**".

<Ctrl-M> the Carriage-Return character normally occurs just before the Line-Feed in Windows/DOS text files; in this case it is considered part of the "newline" and not displayed. With UNIX text files, a Carriage-Return character is displayed as "**<CR>**". With Mac text files, a Carriage-Return is the true "newline" character at the end of each line.

<Ctrl-@> the Null character is normally displayed as any other control character. However, when displayed literally, it is indistinguishable from a space. Alternatively, a visible "Null" character can be chosen with **{CONFIG, Characters/Cursors, Null display character}**. For example, a good value might be "7". Of course, Null would then display the same as Ctrl-G (value 7), but this is usually better than displaying it as a space.

Display Modes

The configuration parameter **{CONFIG, Characters/Cursors, Screen display options}** determines how control and graphics characters are displayed. It can also enable the Hexadecimal, Octal and EBCDIC/custom display modes.

Although this parameter has many possible values, these nine values are the most useful:

0 (Default) Display graphics and control characters literally. "Null" (value 00) characters are displayed according to **{CONFIG, Characters/Cursors, Null display character}**.

1 Display graphics characters literally, but display control characters in the "^*x*" format.

2 Display control characters literally, but display graphics characters as decimal values in the format <*nnn*>.

3 Display graphics characters in the "<*nnn*>" format and control characters in the "^*x*" format.

4 Display all characters literally, including <Tab>, <CR>, <LF> and <Null>. (This mode is used in the ASCII window following **{VIEW, Toggle Hex-mode split}**).

8 Display all characters in hexadecimal. (This mode is used in the Hexadecimal window following **{VIEW, Toggle Hex-mode split}**).

16 Display all characters in octal.

64 Process characters with ANSI/OEM translation.

128 All text characters are processed by the EBCDIC or custom translation table; the resulting characters are displayed literally. The default translation table is EBCDIC — instead of displaying each byte as a normal ASCII character, it is displayed as the equivalent EBCDIC character. The text in EBCDIC mainframe files will then be readable. However, normal ASCII files will display as gibberish.

The default EBCDIC table can be replaced with a custom translation table via **{BLOCK, Edit/translate, Load translate table}**.

As a convenience, **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar) toggles the current window through these nine modes.

In hexadecimal or octal mode, new text must also be entered in the same mode. For example, in hex mode, only hexadecimal digits are valid.

In EBCDIC mode, the ASCII keyboard characters will be translated to the EBCDIC equivalent and entered into the file. Therefore, when you type "A", you will see "A" on the screen, even though the EBCDIC equivalent was entered into the text.

See Also:

The topic "Hex Mode Editing" later in this chapter.
The topic "Translating a Block or File" later in this chapter.

The <Tab> Key and Tab Characters

The **<Tab>** key is almost always 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 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.

When the cursor is within a highlighted block, the operation of **[TAB CHARACTER]** and **[BACKTAB]** changes to indent and undent the entire block.

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

- The tab stops are normally set to every eighth column, but can be changed with **{CONFIG, Tab/fill, Tab stops}**. Tab stops can be set at any desired columns, e.g. at 7, 20, 30 and 73. Tab characters past the last tab stop are displayed as normal control characters, i.e "**^I**".
- 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 whose tab stops are not the same as VEDIT's. This option is also handy with FORTRAN and COBOL programs. It is selected with **{CONFIG, Emulation, Expand <Tab> key with spaces}**.
- Instead of displaying Tab characters using spaces, you can select another character with **{CONFIG, Characters/Cursors, Tab display character}**. A suitable value on an IBM PC is "07". This can make it easier to distinguish between Tab characters and spaces in your file.
- The operation of the <Tab> key according to "insert" mode and within highlighted blocks can be changed with **{CONFIG, Emulation, [TAB CHARACTER] emulation mode}**.

Lower and Upper Case Conversion

This topic is primarily applicable to 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 these values:

0 (Default) No conversion takes place.

1 All lower case letters are converted to upper case. This is similar to the "Caps Lock" on a keyboard.

2 Conditional key conversion — lower case letters are converted to upper case only when the cursor is to the left of the "key conversion character" which is typically ";".

3 Similar to (2) except that characters are reversed instead of being forced to upper case.

4 All upper case letters are converted to lower case.

5 Similar to (2) except that characters are converted to lower case.

Modes "2" and "3" are specifically designed for assembly language programmers who 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 *key conversion character*, typically "**;**". To the right of the "**;**" they are not converted. In this mode an assembly language program can be entered 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 "key conversion character" may be changed with**{CONFIG, Programming, Key conversion character}**.

Mode "3" is similar to Mode "2"; however, 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).

NOTES: This case conversion option does not affect any existing text; use the **{BLOCK, Edit/Translate}** menu to convert existing text.

These modes only affect characters entered in Visual Mode, they do not affect characters entered into dialog boxes.

(DOS Only) **{CONFIG, Misc, Keyboard input options}** permits reversing the case of all typed letters. This affects Visual Mode, dialog boxes and Command Mode.

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}** sub-menu. They are fully described in Chapter 6 (Menu Reference) and in the on-line help for that sub-menu.

The emulation modes include:

- Cursor positioning modes — control how the cursor keys move the cursor past the ends of lines.
- **[TAB CHARACTER]** emulation mode — controls how **<Tab>** and **<Shift-Tab>** work in "Insert" and "Overstrike" modes. Also whether these keys perform Indent/Undent when the cursor is within a highlighted block.
- **<Enter>** key emulation mode — controls how the **<Enter>** key works in "Insert" and "Overstrike" modes.
- **[BACKSPACE]** emulation mode — controls whether this function is destructive in Overstrike/Insert mode, and how it behaves at the beginning of a line.
- Line emulation mode — controls 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.
- Special emulation modes — control how the **[SCREEN BEGIN]**, **[SCREEN END]**, **[SCROLL UP]**, **[SCROLL DOWN]**, **[SEARCH]** and **[DELETE]** functions work.

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

More sophisticated emulation can be performed with the use of keystroke macros and the VEDIT PLUS macro language.

Other Keyboard Input Options (DOS Only - Technical)

Configuration parameter **Config(H_KEY_IN)** (Chapter 8) controls whether VEDIT reads the IBM PC keyboard via "ROM BIOS" or "System". The default is "ROM BIOS" and is only very rarely changed.

The parameter **{CONFIG, Misc, 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 an IBM PC. Note that this has no effect on 8-bit characters already in the file.
- 8-bit characters are normally treated as text (graphics) characters. Alternatively, they can be treated as function keys. This is not desirable on an IBM PC; it is desirable when running VEDIT on external CRT terminals (UNIX version).
- Unassigned function/control keys can enter their codes directly into the text. This is not desirable on an IBM PC; 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 many years ago some users asked for it.)
- In a key-sequence, the 2nd and following Ctrl character can be converted to the equivalent letter. This is useful for the WordStar emulation keyboard layout so that e.g. ^K ^V is equivalent to ^K V. Since this does not affect other layouts, it is the default.

Other Screen Display Issues (DOS Only - Technical)

On the IBM PC, you can select from seven different cursor types — four "software" cursors and three "system" cursors. VEDIT is typically configured to have a different cursor appearance in "Insert" versus "Overstrike" mode. For the software cursors, you can select the blink rate or a non-blinking cursor and a specific cursor color. For the system cursors, you can choose a thin, medium or full-height cursor. These options are a matter of personal preference and are configured with the **{CONFIG, Characters/Cursors}** sub-menu.

(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.

Hex Mode Editing (and Octal)

For some types of editing, particularly "binary" files, it is easiest to display and edit the file in hexadecimal. Octal editing is also available.

You can switch to hexadecimal (or octal) editing in three ways:

- Select **{VIEW, Toggle hex mode split}** (<**Alt-=**>). This is usually the preferred way.
- Select **{VIEW, Toggle display mode}** (<**Alt-D**> or toolbar) several times until the window is in hexadecimal or octal.
- Set **{CONFIG, Characters/Cursors, Screen display mode}** to "8" (or "16" for octal).

{VIEW, Toggle hex mode split} lets you edit a file in two windows; one displays in hexadecimal, the other in normal ASCII. (You can also display in EBCDIC.) The ASCII window initially uses display mode "4" in which all characters, including the "newline" <LF>, <CR> and <Tab> characters are displayed literally (using the IBM PC character set). The cursors in both windows are synchronized and will move together.

HINT: When editing in hexadecimal, you may find it easier to treat the file as a binary file with a uniform 16 or 64 characters displayed per screen line. You can easily enter Binary-16 mode by selecting **{VIEW, Toggle hex-mode split}** again. **{VIEW, Toggle binary/text mode}** (**<Alt—>**) toggles between Binary-16, Binary-64 and the normal file types.

NOTE: By default, selecting a binary file type will only let you overstrike characters and not insert or delete characters. If you need to insert/delete, change **{CONFIG, File handling, Overwrite mode}** to "0".

Use **{WINDOW, Next window}** (<**Ctrl-F6**>), **{WINDOW, Switch}** (<**Alt-F5**>) or the mouse to switch between the windows. If desired, use **{VIEW, Toggle display mode}** (<**Alt-D**>) to toggle either window to a different display mode. For example, you can toggle the ASCII window to EBCDIC so that you can edit in hex and EBCDIC at the same time.

➢ **To edit a file in split-screen Hex and ASCII with 16 characters per line:**

1. Select **{VIEW, Toggle hex mode split}**, default: <**Alt-=**>.
2. Select it again to toggle to Binary-16 mode.

In a hex-mode window, new characters must be entered in hexadecimal, i.e. by entering "00" thru "FF". Other characters cause an error beep. Similarly, in octal-mode, new characters must be entered in octal, i.e. by entering "000" thru "377".

When the current window is displayed in hexadecimal, the status line changes to display the cursor's hexadecimal offset into the file.

Searching in Hex, Decimal or Octal

You can search for hexadecimal, octal or decimal values. With normal pattern matching use "|**H***hh*" to search for hex value '*hh*' Similarly, use "|**O***ooo*" to search for octal value '*ooo*'. Use "|*ddd*" to search for decimal value '*ddd*'.

|**H***hh* Match the character with hex value '*hh*'. Both digits must be present. Each hex value must be preceded with "|**H**"

|**O***ooo* Match the character with octal value '*ooo*'. All three digits must be present.

|*ddd* Match the character with decimal value '*ddd*'.

Examples:

|h74|h68|h65	Search for the word "the" in hex.
|o164|o150|o145	Search for the word "the" in octal.

NOTES: "**|**" is the "pipe" character, which is <Shift>-**** on the keyboard.

When you enter a letter in hex, decimal or octal, the search automatically becomes case sensitive.

Since it is tedious to precede hex values with "|**H**", you can alternatively select the "**( ) Hex**" search mode. In this mode, all search characters are entered as two-character hex codes, "00" thru "ff", followed by a space. This is identical to the way characters are displayed in VEDIT's hex mode. For example, to search for "ABC123", enter:

41 42 43 31 32 33

Alternatively, hex words, double-words and quad-words can be entered without spaces, but with an optional ":". The most significant byte is entered first. Therefore, the following are equivalent:

0a496e79 23fa45e8

0a49:6e79 23fa:45e8

79 6e 49 0a e8 45 fa 23

NOTE: In Hex mode, only simple searches, without pattern matching, are supported. The search is also case sensitive.

See Also:

On-line help for the Search dialog box.

Entering Numbers in Hexadecimal

At any prompt for a number, e.g. **{GOTO, Line #}**, you can enter a hexadecimal value by preceding it with "0x" or "0h". Enter an octal value by preceding the octal digits with "0o". You can also enter numeric expressions and mixed-radix expressions.

Example of a mixed-radix numeric expression:

(0x3A6FF + 1000) / 2

Undo and Redo

VEDIT remembers each edit operation performed and can quickly undo them in reverse order, including cursor movements, insertions, deletions and search/replace. You can undo these operations keystroke-by-keystroke, line-by-line or deletion-by-deletion. This not only lets you back out of mistakes, but also lets you try out changes and "undo" them if they don't work out.

Sometimes when you use Undo, you accidentally go too far and remove changes you wanted to keep. Or you simply change your mind and want to "undo" the undo. This capability is called "redo".

VEDIT normally remembers the last 1000 edit operations in ***EACH*** file being edited. As you continue to work and exceed the maximum number of edit operations VEDIT can remember, oldest operations are forgotten. Because of this, you can only go as far back as 1000 keystrokes.

Notes:

Each basic edit operation uses one "undo level"; VEDIT is typically configured for 1000 undo levels. Therefore, VEDIT could undo the last 1000 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), 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 cannot be undone.

Two configuration parameters **Config(U_UNDO_MAX)** and **Config(U_UNDO_MIN)** determine how many undo levels are available to each file. When sufficient memory is available, 1000 levels are typically available. As memory becomes tight as more files are opened, new buffers may only have the minimum number of levels available; this is typically 100.

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 256K (64K for 16-bit DOS version) 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:

```
Cannot undo this operation! Proceed anyway?
                [Yes] [No]
```

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:

{EDIT, Undo} menu in Chapter 6 (Menu Reference).

File Types - Win/DOS, UNIX, Mac, Binary

VEDIT can edit both text files and binary/data files. A program source code file or the chapter of a manual are typical text files, while executable files (.EXE) database files (.DBF) are typical binary/data files.

With VEDIT, the file type is mostly a matter of the "newline" character(s) used in the file. For Windows/DOS and UNIX text files, VEDIT expects Line-Feed characters to end each text line; for Mac text files, Carriage-Return characters end each text line. If the Line-Feed characters in a text file are preceded with Carriage-Return characters, VEDIT considers the file to be a Windows/DOS text file, otherwise it's a UNIX text file. If only Carriage-Return characters are found, it's a Mac text file.

If no or very few "newlines" are found, e.g. the lines are more than 2000 characters long, VEDIT considers the file to be a binary/data file.

The difference between Windows/DOS, UNIX and Mac text files is important in the way that "newline" characters are displayed, deleted and inserted.

When opening a file for editing, VEDIT examines the file to automatically determine the file type. Sometimes VEDIT will choose the wrong file type. For example, since a typical executable file (.EXE) contains random Line-Feed characters, VEDIT will usually treat it as a text file.

If you set **{CONFIG, File handling, Enable auto-file type}** to "No", VEDIT will not automatically determine the file type when it opens a file.

Binary/data files are displayed with a uniform number of characters per line. This also handles "fixed-length record" database files. For these files, **{CONFIG, File handling, File type}** sets the number of characters displayed per line, i.e the "record size". "64" is the default for binary files; you may want to change it to "16" when editing in hexadecimal.

{CONFIG, File handling, File type} lets you see and/or change the current file's type. Each file opened can have its own setting.

0=CR+LF	Each text line ends in both a CR (Carriage-Return) and LF (Line-Feed). Typical for Windows/DOS.
1=LF	Each text line ends in just a LF. Typical for UNIX/QNX.
2=CR	Each text line ends in just a CR. Typical for Macintosh.
n=recsize	Record Mode. Values of "8" through "2048" correspond to the record length used for fixed-length-record data files. Instead of assuming that lines end in a "newline" character, VEDIT treats each line (record) as simply 'n' characters. Word processing operations are not available in Record Mode.

Windows/DOS and UNIX Text Files

Each line in a Windows/DOS text file normally ends in both a Carriage-Return and Line-Feed character; this **<CR><LF>** pair is considered the normal "newline" character even though it really is two characters. Pressing **[DELETE]** once at the end of a line deletes the **<CR><LF>** pair. Similarly, pressing **<Enter>** (in Insert Mode) inserts a **<CR><LF>** pair.

If a text line ends in just a Line-Feed when the file type is set to "0=DOS text", "**<LF>**" is displayed at the end of the line. Such a line may be corrected by deleting the lone **<LF>** with **[DELETE]** and then inserting the **<CR><LF>** pair with **<Enter>**.

UNIX text files normally have lines ending in just a Line-Feed character, which is therefore the "newline" character. Pressing **<Enter>** inserts only the Line-Feed character. Carriage-Return characters have no special meaning and are displayed as "**<CR>**".

VEDIT does not automatically convert a Windows/DOS file to UNIX or vice versa. However, you can easily convert an entire file or just a highlighted block.

- **To convert a Windows/DOS file into a UNIX file:**

  1. To convert the entire file, select **{BLOCK, Select all}** (**<Ctrl-A>**) to block highlight the file.
  2. Select **{BLOCK, Convert newlines, DOS to UNIX}**. The block/file is now converted.

- **To convert a UNIX file into a Windows/DOS file:**

  1. To convert the entire file, select **{BLOCK, Select all}** (**<Ctrl-A>**) to block highlight the file.
  2. Select **{BLOCK, Convert newlines, UNIX to DOS}**. The block/file is now converted.

Macintosh Text Files

Mac text files use a single "Carriage-Return" as the "newline" character at the end of each line. Therefore, pressing **<Enter>** (in Insert Mode) inserts a **<CR>**. Line-Feed characters have no special meaning and are displayed as "**<LF>**".

You must use additional care when editing Mac files. In particular you must be very careful with cut/paste operations between Mac and Windows/DOS files. For example, a multiple line block cut from a Mac file will become a single long line when pasted into a Windows/DOS file.

NOTE: Because "regular expressions" were originally designed for UNIX text files with Line-Feed "newline" characters, you may have some trouble searching Mac files using some regular expressions.

VEDIT does not automatically convert from one file format into another. Therefore, before performing cut/paste between Mac and Windows/DOS files you may want to convert the Mac file into a Windows/DOS file.

➢ **To convert a Mac file into a Windows/DOS file:**

1. To convert the entire file, select **{BLOCK, Select all}** (**<Ctrl-A>**) to block highlight the file.
2. Select **{BLOCK, Convert newlines, Convert macro}**. The macro will display a menu of conversion choices.
3. Select the "Mac to DOS" conversion by typing the number for this choice. The block/file is now converted.

Binary/Data Files (Record Mode)

VEDIT treats files that do not have any "newline" characters in them as binary/data files. Files with text lines longer than about 2000 characters are also treated as binary/data files.

Binary/data files are usually edited in "Record Mode" in which a uniform number of characters are displayed per line or "record". This defaults to 64, but can be changed with **{CONFIG, File handling, File type}**. Selecting a value of "8" - "2048" selects Record Mode and selects the number of characters displayed per line. The number of characters per line is referred to as the "record size".

Setting the desired record size also handles files with "fixed-length records". For example, files downloaded from IBM mainframes often have records of 80 or 256 characters. Most database files also have fixed-length records.

The record size is often greater than the width of the current window. The record can be viewed using horizontal scrolling or by wrapping the record onto multiple screen lines. **{CONFIG, Display options, Horizontal scroll margin}** controls how long lines are displayed.

In Record Mode, any Line-Feed and Carriage-Return characters are no longer treated as "newline" characters. However, they may display as "<LF>" and "<CR>". Similarly, control and graphics characters are displayed according to the current display mode. Repeatedly selecting **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar) will let you find the desired display mode.

NOTES: When editing a binary/data file, you may want to use **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar) to select "ASCII-4" which displays any Line-Feed, Carriage-Return, Tab and other control characters as a single character instead of as **<LF>**, etc.

Binary/data files are often most easy to edit with a split window displaying hexadecimal and ASCII. This can be selected with **{VIEW, Toggle hex-mode split}** (**<Alt-=>**).

By default, you can only overstrike characters in "Record mode"; you cannot insert or delete characters. However, full editing can be enabled by setting **{CONFIG, File handling, Overwrite-only mode}** to "0".

Database Files With Headers

Some data files, particularly database files, consist of fixed-length records that follow a variable length header. As described above, to edit these files **{CONFIG, File handling, File type}** should be set to the length of the records. However, since the header is usually not the same length as the records, the beginning of each record will not be displayed at the beginning of a screen line and the fields in the records will not be lined up.

You can make these files easier to edit by setting **{CONFIG, File handling, Record header size}** to the size of the header in bytes. Each following record will then start on a new screen line and the fields in the records will be aligned on the screen.

The status line displays a line number of "0" when the cursor is in the header. This makes the line number correspond to the record number.

xBASE Files: VEDIT can configure itself to dBase and xBase database files by selecting **{MISC, More macros, DBASEKEY}**. This macro reads the header size and record size from the current ".DBF" file and configures VEDIT accordingly.

A more elaborate DBASE.VDM macro is described in Chapter 5.

Editing Very Long Lines (Technical)

VEDIT cannot always handle text "lines" longer than about 2500 characters. Specifically, the first character displayed in the window through the last character of the last line of the window cannot exceed the size of the edit buffer. Therefore, assuming a 24 line window, a typical buffer size of 60,000 bytes and uniform length lines, the maximum line length is 60,000/24 = 2666.

For this reason, VEDIT automatically sets the file type to "Record Mode" when it encounters lines longer than 2000 characters at the beginning of a newly opened file. In Record Mode, VEDIT can handle lines of any length, even millions of characters long.

Other than editing in Record Mode, you can also get around the 2500 character line length limitation for text files in several ways:

- If most lines are short and only a few are very long, VEDIT should have no trouble editing the file in text mode. Even a few lines up to 30,000 characters can be edited.
- Use **{CONFIG, Display options, Horizontal scroll margin}** to wrap very long lines onto multiple screen lines. For example, with the maximum suggested value of 2048, a 10,000 character long line would be wrapped onto five screen lines.
- Edit the file in a smaller window. For example, a 12 line window could edit uniform length lines of 5332.

NOTE: The upcoming 32-bit version of VEDIT 5.2 will be able to directly edit much longer text lines.

Keystroke Macros

You can assign a frequently typed sequence of keystrokes to a single function/control key. Pressing the single function/control key then performs the equivalent of typing the entire sequence of keys. This saves time, effort and reduces the chance of error. These stored keystroke sequences are called "*keystroke macros*". You can define numerous keystroke macros and each may contain up to several hundred keystrokes.

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 "normal" and other supplied keyboard layouts. The menus display any "hot-keys" that directly access each item.
- Keystroke macros can access the VEDIT PLUS macro language. Although this manual does not describe the macro language in detail, Appendix C and the file KEY-MACRO.LIB list many 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 **<Alt-Q>** to type out the phrase "attached and included herein by reference" each time it was pressed.

You can add new keystroke macros at any time. Each new keystroke macro is normally assigned to a function or control key that is not already in use; these available function/control keys can be displayed with **{CONFIG, Keyboard layout, Display unused keys}**.

Keystroke macros can be added in several different ways:

- **{CONFIG, Keyboard layout, Record keystroke macro}** lets you record a new keystroke macro as you edit the file.
- **{CONFIG, Keyboard layout, Add Keystroke macro}** lets you define a new keystroke macro using a dialog box which records your keystrokes without making any edit changes. Limited editing is provided.
- **{CONFIG, Keyboard layout, Edit/view layout}** lets you edit the entire keyboard layout, including any keystroke macros, as a normal text file. You can cut and paste between layouts, etc. These changes can be temporary, or can be permanently saved in the **`vedit.key`** file.

  The topic "Editing the Keyboard Layout" describes this in detail.
- You can directly edit the **`vedit.key`** file and make any desired changes. This is similar to **{CONFIG, Keyboard layout, Edit/view layout}**.

  Chapter 8 (Configuration) describes this in detail.

NOTE: VEDIT forgets new keystroke macros when you exit unless you make them permanent with **{CONFIG, Keyboard Layout, Save layout}**.

Recording Keystroke Macros

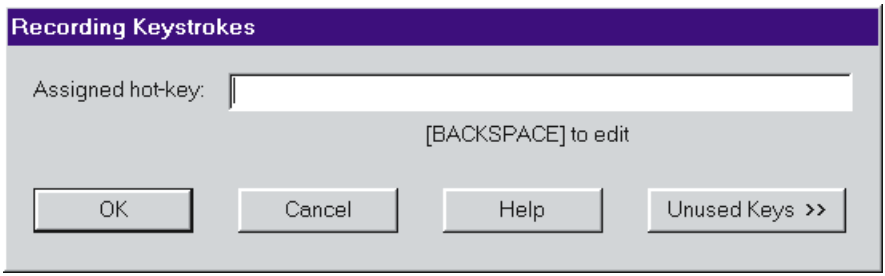
You can record a new keystroke macro while you perform a multi-step editing operation. Afterwards, you only have to press the hot-key to repeat (play back) the entire editing operation.

The following (contrived) example demonstrates how to use **{CONFIG, Keyboard layout, Record keystroke macro}** to record a keystroke macro. It performs the following steps:

1. Go to the beginning of the current line and type “Begin:”.
2. Go to the end of the line and type “:End”.
3. Advance to the next line.

➢ **Example of recording a keystroke macro:**

1. Select **{CONFIG, Keyboard layout, Record keystroke macro}** (<**Alt-K**>). You will see the following dialog box:



2. At the “Assigned hot-key:” prompt, press the desired “hot-key” for the keystroke macro. For this example, press <**Alt-Q**>.

   If you make a mistake, press <**Backspace**> to delete the keystroke.

   Press <**Tab**> to accept “**Alt-Q**”. If the key is already in use, you will be prompted for confirmation to overwrite it.

3. Assuming the “Normal” keyboard layout, press the following keys to go to the beginning of the current line and type “Begin:”.

   <Home> B e g i n :

   Then press the following keys to go to the end of the current line, type “:End” and advance to the next line.

   <End> : E n d <Ctrl-Enter>

4. Turn off (stop) the Record Macro mode by pressing the key indicated on the status line. With the “Normal” layout it is <**Alt-K**>. The new keystroke macro is now fully defined.
5. Test the new keystroke macro by pressing its hot-key <**Alt-Q**>.
6. To make new keystroke macro permanent, select **{CONFIG, Keyboard layout, Save layout}** to save the layout into the **`vedit.key`** file.

Adding Keystroke Macros

{CONFIG, Keyboard layout, Add Keystroke macro} lets you define a new keystroke macro using a dialog box that records your keystrokes without making any changes to the files being edited. Experienced VEDIT users will prefer it over "Record keystroke macro" for two reasons:

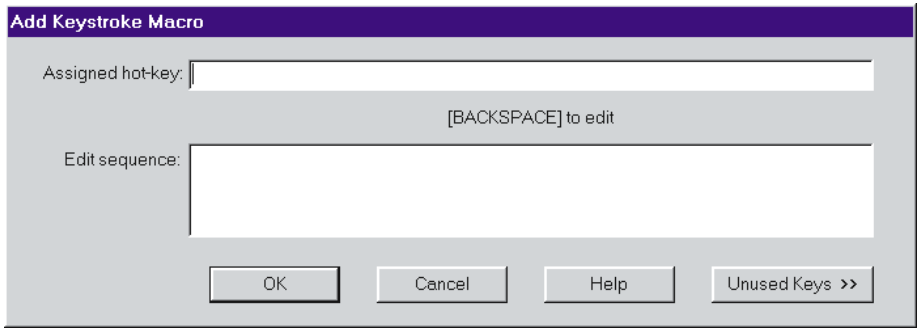
- It provides simple editing in case you make a mistake.
- You can add keystroke macros that access the VEDIT PLUS macro language. The supplied KEY-MAC.LIB provides many examples.

The following example creates the same keystroke macro as above.

➢ **Example of adding a keystroke macro:**

1. Select **{CONFIG, Keyboard layout, Add keystroke macro}** (**<Alt-A>**).
2. At the "Assigned hot-key:" prompt, press the desired "hot-key" for the keystroke macro. For this example, press **<Alt-Q>**.

   Press **<Tab>** to accept **<Alt-Q>**. If the key is already in use, you will be prompted for confirmation to overwrite it.



3. At the "Edit sequence:" prompt, enter the same sequence of keys as for the previous example.

   Notice that when you press **<Home>**, "[LINE BEGIN]" is displayed. VEDIT records edit functions by name and not by their currently assigned keys. (See Notes: below.)

   If you make a mistake, press **<Backspace>** to delete the keystroke. Finally, press **<Enter>** to accept the edit sequence.
4. Test the new keystroke macro by pressing its hot-key **<Alt-Q>**.
5. To make new keystroke macro permanent, select **{CONFIG, Keyboard layout, Save layout}** to save the layout into the **`vedit.key`** file.

Notes:

{CONFIG, Keyboard layout, Edit/view layout} displays and lets you edit all active keystroke macros. The **<Alt-Q>** defined in the example above would display as:

```
Alt-Q   [LINE BEGIN] Begin:[LINE END] :End[NEXT LINE]
```

You can then select **{FILE, Print}** to create a printout of the entire keyboard layout.

VEDIT records the edit functions in a keystroke macro by name and not by their currently assigned keys. This lets you change the keyboard layout without having to re-enter existing keystroke macros. In the example assignment to "<Alt-Q>" above, if you redefined the **<Home>** and **<End>** keys, the keystroke macro would still work.

See Also:

{CONFIG, Keyboard layout} in Chapter 6 (Menu Reference).

Adding Keystroke Macros from KEY-MAC.LIB

The file KEY-MAC.LIB includes many keystroke macros that you may find useful. Although the VEDIT PLUS macro language is used in the keystroke macros, they will also work in VEDIT.

NOTES: Keystroke macros that use the VEDIT PLUS macro language can be added with "Add keystroke macro", but not with "Record keystroke macro".

In practice it is easier to cut & paste keystrokes from KEY-MAC.LIB into the keyboard layout with **{CONFIG, Keyboard layout, Edit/view layout}**. The topic "Editing the Keyboard Layout" gives a step-by-step example.

➢ **Example - Add a keystroke macro from KEY-MAC.LIB:**

This example shows how to add a typical keystroke macro from KEY-MAC.LIB to VEDIT. In this case, the macro that inserts the current date and time will be assigned to **<Ctrl-F12>.**

In practice, you can open the file KEY-MAC.LIB in VEDIT and scroll the screen until the desired macro is visible in the lower half of the screen. The macro is listed in KEY-MAC.LIB as:

```
[VISUAL EXIT]
Out_Ins( ) Date(NOCR) Ins_Text("  ") Time(NOCR)
Out_Ins(CLEAR)
```

1. Select **{CONFIG, Keyboard layout, Add keystroke macro}** (or press **<Alt-A>**).
2. At the "Assigned hot-key:" prompt, press **<Ctrl-F12>**. Then press **<Tab>** to accept **<Ctrl-F12>.**
3. At the "Edit sequence:" prompt, first press the key that is currently assigned to **[VISUAL EXIT]**; this is typically **<Ctrl-E>**. The screen will

echo "[VISUAL EXIT]". ***DO NOT*** type the characters "[VISUAL EXIT]".

DO NOT PRESS **<Enter>**. KEY-MAC.LIB lists the macros on multiple lines for clarity and because some of them are quite long. Keystroke macros are entered as one long line. The only exception is if the macro contains a "[RETURN]"; in its place first press **[ENTER CTRL]** (**<Ctrl-Q>**), then press **<Enter>**.

Enter the rest of the keystroke macro just as it appears in KEY-MAC.LIB. If you make a mistake, press **[BACKSPACE]** to delete the keystroke. Finally, press **<Enter>** to accept the edit sequence.

4. To make new keystroke macros permanent, select **{CONFIG, Keyboard layout, Save layout}** to save the layout into **`vedit.key`**.

Modifying an Existing Keystroke Macro

While it is usually better to use **{CONFIG, Keyboard layout, Edit layout}** to modify a keystroke macro, it can also be done with the "Add keystroke macro" function.

➢ **Modifying an existing keystroke macro**

1. Select **{CONFIG, Keyboard layout, Add keystroke macro}**.
2. Press the "Function/Control Key" assigned to the keystroke macro you want to modify. Press **<Tab>**.
3. In response to "Redefine existing key? [Yes] [No]", select **[Yes]**.
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. Then use **[BACKSPACE]** to erase part of the macro and/or type in additional keystrokes. When finished, press **<Enter>**.

In effect, VEDIT does not "forget" the keystroke macro until you are done defining the new one. This is helpful, not only for editing existing macros, but also for building a larger keystroke macro from smaller ones.

Deleting Keystroke Macros

Unused keystroke macros can be deleted by selecting **{CONFIG, Keyboard layout, Edit/view layout}** and deleting the corresponding line. Or you can use the "Add keystroke macro" function.

➢ **To 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 **<Enter>**. Since the key is now assigned to "nothing", it is removed from the list of assigned keys.

Editing the Keyboard Layout

{CONFIG, Keyboard layout, Edit/view layout} lets you edit the current keyboard layout as a normal text file. When done, the new layout is automatically loaded, and can optionally be made permanent by saving it as the **`vedit.key`** file.

➢ **To edit the keyboard layout:**

1. Select **{CONFIG, Keyboard layout, Edit/view layout}**.

   This saves the current keyboard layout into the file "veditkey.tmp" and opens the file for editing.

2. Edit the keyboard layout as desired. Follow these guidelines:

   A. Do not move or alter the first line which assigns the <Enter> key to the function [RETURN].

   B. Notice that all lines have the same format. Each line begins with the key or keys that 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.

   C. 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" (i.e. a "Carriage-Return, Line-Feed") enter "[RETURN]".

3. Press **[VISUAL EXIT]** or **<Ctrl-E>** when done to save or abandon your changes.

4. At the prompt, select whether you want to ignore (abandon) or save your changes:

   [I]gnore, [T]emporary, [S]ave layout into VEDIT.KEY?

   [I] - Your changes are ignored. The layout is not changed.

   [T] - The layout takes effect, but is temporary - it will be lost when you exit VEDIT.

   [S] - The layout takes effect and is saved as the **`vedit.key`** file, just as if you selected **{CONFIG, Keyboard layout, Save layout}**.

   After your selection, you are returned to your original file.

If VEDIT detects an error in the edited keyboard layout, it reports an error with the line number and gives you a choice of editing the layout again or quitting the function.

NOTE: A common error is to forget a "newline" on the last line of the keyboard layout.

See Also:

{CONFIG, Keyboard layout, Edit/view layout} in Chapter 6 (Menu Reference).

Adding a Keystroke Macro from KEY-MAC.LIB

NOTE: This topic assumes you already know how to open and close files, edit multiple files, and cut and paste blocks of text.

The file KEY-MAC.LIB is a library of useful keystroke macros that can be added to VEDIT. Included are macros for moving the cursor by sentences, counting words in a file, a window color selection chart and much more. Although the VEDIT PLUS macro language is used in the keystroke macros, they will also work in VEDIT.

➢ **To add a keystroke macro from KEY-MAC.LIB:**

1. Select **{CONFIG, Keyboard layout, Edit layout}** so that you can edit the current keyboard layout as the file "veditkey.tmp".
2. Open the file "key-mac.lib" for editing. It should be in the *VEDIT Home Directory*.

   E.g. use **{FILE, Open}** (**<Ctrl-O>**) to open the file **`c:\vedit\key-mac.lib`**
3. Highlight the macro to be added. Most macros begin with "[VISUAL EXIT]". Then copy the macro to the scratchpad (text register "0").
4. Toggle back to the file "veditkey.tmp". You can use **{FILE, Next buffer}** (**<F6>** or toolbar).
5. Go to the end of the keyboard layout and insert the scratchpad (text register "0") containing the keystroke macro.

   Edit it so that the entire macro appears on one line. Use other entries in the layout as a guide. Be sure the line ends in a "newline" and that there are no extra blank lines in the file.
6. At the beginning of the line, type the name of the key to which the macro is assigned, e.g. "**Ctrl-F12**".

   You may want to perform a search for the key name to be sure it is not already assigned to something else.
7. Press **[VISUAL EXIT]** or **<Ctrl-E>** when done editing the layout.
8. At the prompt, select whether you want to ignore (abandon) or save your changes.
9. If desired, switch back to the file "key-mac.lib" and close it.

Notes:

A keystroke macro must begin with **[VISUAL EXIT]** (**<Ctrl-E>**) to enter Command Mode. When done, the keystroke macro will automatically return to normal editing. (VEDIT PLUS users: a final "V" command is not needed and should even be avoided.)

The upper limit for the length of a single keystroke macro is 4000 characters. Complex macros that display menus, etc., should be loaded into text registers.

Macro language commands can also be mixed with normal edit functions and Menu selections. Use **[VISUAL EXIT]** to switch to Command Mode. VEDIT 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.

Loading a New Keyboard Layout

You can easily load a different keyboard layout, e.g. WordStar or a custom layout, without re-installing.

In general, you can change the entire keyboard layout "on the fly" and load different sets of keystroke macros for different editing tasks. This is also useful when several people use a computer and would like different keyboard layouts. This is described under **{CONFIG, Keyboard Layout, Load layout}** in Chapter 6 (Menu Reference).

➢ **Example - Load the WordStar keyboard layout:**

1. Select **{CONFIG, Keyboard layout, Load layout}**.
2. Change the default filename of "vedit.key" to "wordstar.key" and press **<Enter>**. Alternatively, change the filename to "*.key" for point and shoot file selection. The new keyboard layout will be loaded.
3. To make the new keyboard layout permanent, select **{CONFIG, Keyboard layout, Save layout}** to save the layout into the **`vedit.key`** file.

Notes:

You could also change the keyboard layout by simply copying "wordstar.key" or "brief.key" to "vedit.key".

You can load a keyboard layout when you start up VEDIT. For example, the command to load the WordStar keyboard layout on startup is:

```
vedit -c"key_load('wordstar.key')"
```

Therefore, you could create different VEDIT icons, each with a different keyboard layout. (See the topic "Changing the VEDIT Icon Properties".)

The **`startup.vdm`** file shows how to define custom keystroke macros at startup; these will override any assignments in the **`vedit.key`** file.

Block Operations

VEDIT has an exceptionally wide range of block operations. Not only can you copy, move and delete blocks, you can also translate blocks, fill blocks with any desired character, indent blocks and insert an empty block between two columns. A "*block*" can be any amount of text from one character to an entire file.

VEDIT supports three types of blocks:

- **Stream block**. This is a block of contiguous characters. You precisely determine the characters in the block. For example, it can be all characters from the middle of one line to the middle of another line.
- **Line block**. This block consists of entire lines including the "newline" at the end of each text line. (For fixed-length records, it includes all characters in the record.)
- **Columnar block**. This block is a rectangle of characters in your text, i.e. only those characters that are within the specified columns.

Stream blocks are the default and most commonly used type of block. You can think of line blocks as simply a shortcut way of selecting a stream block that consists of entire lines. However, columnar blocks are quite different from stream blocks and VEDIT has many special columnar block features.

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

Block editing functions such as **{BLOCK, Write to disk}** are not available until you have selected a block of text.

Selecting a block of text changes the operation of some editing functions (the cursor must also be within the block):

- **[DELETE]** deletes the entire block of text. (This is configurable.)
- **{EDIT, Indent}** and **{EDIT, Undent}** functions indent/undent all lines in the block.
- **[TAB CHARACTER]** and **[BACKTAB]** are equivalent to **{EDIT, Indent}** and **{EDIT, Undent}**. (This is configurable.)
- **{SEARCH, Search}** and **{SEARCH, Replace}** by default are restricted to the characters in the block. This can be selected in the Search and Replace dialog boxes.

Marking (selecting) a Block of Text

There are three main ways to mark a block of text:

- Select the **{BLOCK}** menu items or the equivalent "hot-keys" to mark the desired type of block. Stream and columnar blocks can also be set from the toolbar.

- Hold down the **<Shift>** key while moving the cursor to mark a stream block. You can use the cursor keys and almost any cursor movement function. For example, **<Shift><Ctrl-End>** marks from the current position to the end of the file.
- Use the mouse to "drag" over the desired stream block. By also holding down the **<Alt>** key, it marks a columnar block.

  To highlight a large block, first highlight a small section at the beginning of the block; then go to the end of the block; finally, hold down the **<Shift>** key and finish highlighting the entire block. Holding down the **<Shift>** key lets you expand the size of a highlighted block.

Small block are easily marked with the **<Shift>** key or mouse. It sets both the beginning and ending block markers; all block operations are then available. However, the **{BLOCK}** menu items or equivalent hot-keys give you more flexibility and make it easier to mark a very large block.

➢ **To mark a large block of text (assuming no block markers are set):**

1. Position the cursor on the first character to be included in the block. (Or you can mark the end of the block first.)
2. Set the first block marker with **{BLOCK, Set stream marker}** (**<F9>** or toolbar). Note the message "1-END" on the status line.

   Alternatively, select a columnar block with **{BLOCK, Set column marker}** (**<Alt-I>** or toolbar).
3. Move the cursor to the end of the block; any desired method can be used. For example, you can search for text at the end of the block.

   For stream blocks, the cursor should be positioned just past the last character of the block. To include the "newline" at the end of a line, position the cursor at the beginning of the next line.
4. Set the second marker by again selecting **{BLOCK, Set stream marker}** or **{BLOCK, Set column marker}**. Note the message "BLOCK" on the status line.

After marking a block, you can change its size and/or the type of block. Move the cursor to the desired end of the block and then select "Set stream marker", "Set column marker" or "Set line marker" from the **{BLOCK}** menu.

NOTE: The "normal" keyboard layout assigns **<F9>** as a "hot-key" for **{BLOCK, Set stream marker}**. If your layout does not have an assignment for this function, you can alternatively press **{BLOCK, Copy to cursor}** (**<Ctrl-F9>**) to mark a block. Until both block markers are set, "**Copy to cursor**" only sets the block markers; when both block markers are set, it copies the block to the current cursor position.

VEDIT's Blocks are Persistent

VEDIT's blocks are "persistent"; after marking a block, it remains marked until you process it or explicitly remove the markers. Simply moving the cursor doesn't remove the markers. Persistent blocks offer flexibility you otherwise can't have:

- After marking a block, you can move the cursor anywhere and then directly copy/move the block to the new position without using a scratch-pad/clipboard; this saves keystrokes.
- After marking a block, you can easily change its size.
- After marking a block, you can restrict a search/replace to just the block.

Most block operations remove the markers after the operation is completed.

The disadvantage of persistent blocks is that you sometimes have to explicitly remove the markers. VEDIT makes this as easy as possible.

You can remove the block markers in many ways:

- Double-press the **<Ctrl>** key. This assumes **{CONFIG, Emulation, Alt/Ctrl/Shift key shortcut modes}** is enabled.
- Simultaneously press both mouse buttons.
- Press **[CANCEL]** (**<Ctrl-**\>).
- Select **{BLOCK, Remove markers}** (**<Shift-F9>** or toolbar).
- Select **{ESCAPE, Remove block markers}**.
- (DOS) Press **<Ctrl-Break>**.

What exactly does the block include?

For "stream" 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 stream block at the end of a line, the (invisible) "newline" character will not be included. If you end the block at the beginning of the next line, the preceding "newline" character will be included. When marking entire lines, you may find it easier to select "line" blocks.

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.

However, 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.

{BLOCK, Copy / Move to cursor}

{BLOCK, Copy to cursor} and **{BLOCK, Move to cursor}** save steps when performing the common operations of copying or moving a block of text within your file.

Instead of having to first copy or cut the block to the scratchpad/clipboard, you can simply copy/move a highlighted block to the current cursor position. This is possible because VEDIT's blocks are *persistent* — after highlighting a block, you can move the cursor and the block remains highlighted.

{BLOCK, Copy / Move to cursor} can also be used to set block markers. Their operation depends upon how many block markers are set; see the table below. In this way, only a single key is needed to copy a block of text.

➢ **To directly copy/move a block using a single key:**

1. Move the cursor to the first character of the block. Press **<Ctrl-F9>** (the hot-key for **{BLOCK, Copy to cursor}**).
2. Move the cursor past the last character of the block. Press **<Ctrl-F9>** again. The block is now highlighted.
3. Move the cursor to the destination for the block. Press **<Ctrl-F9>** for the third time. The text will be copied in front of the cursor.

   Alternatively, use **<Alt-F9>** to move the block.

Block Function Key Behavior

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

Notes:

The "normal" VEDIT keyboard layout assigns **<F9>** to **{BLOCK, Set stream marker}**. Alternatively you could assign **<F9>** to **{BLOCK, Copy to cursor}**. **<F9>** would then perform a convenient combination of setting block markers and copying a block of text. (Earlier versions of VEDIT did this.)

Copying a block of text to two or more places in your file or into another file is best done with the use of a scratchpad (text registers).

Text Registers and the "Scratchpad"

VEDIT has over 100 text holding areas called "*text registers*", each of which is similar to the "scratchpad" or "clipboard" in other editors. The text registers are numbered from "0" to "100". (Some additional registers above 100 are only accessible in the VEDIT PLUS macro language.) By convention, registers "0" through "9" are reserved for "cut and paste" operations.

The text registers are accessed with **{BLOCK, Copy to register}**, **{BLOCK, Move to register}** and **{BLOCK, Insert register}**. These functions prompt for the register's number. If you simply press **<Enter>** or double-press the function's hot-key, it selects the default register "0", which is also called the "*scratchpad*".

The "Scratchpad" Text Register

Text register "0" is also called the "scratchpad". It is the default "cut and paste" text register when no other is explicitly selected. The scratchpad can also be accessed from the **{EDIT, Scratchpad}** menu. Therefore, **{EDIT, Scratchpad, Copy to scratchpad}** is identical to **{BLOCK, Copy to register}** and selecting the default register "0".

The scratchpad is used for simple cut and paste operations, especially for repeatedly inserting the same block of text into a file or for copying a block from one file to another.

Remember, you can directly copy/move a block within a file without a text register by using **{BLOCK, Copy to cursor}** and **{BLOCK, Move to cursor}**.

Scratchpad Functions

Function	Key	Operation
{EDIT, Scratchpad, Cut}	<Shft-Del>	Move or "cut" to the scratchpad
{EDIT, Scratchpad, Copy}	<Ctrl-Ins>	Copy the block to the scratchpad
{EDIT, Scratchpad, Paste}	<Shft-Ins>	Insert or "paste" the scratchpad

The "normal" hot-keys for the scratchpad are the old-style clipboard keys, before the new-style <Ctrl-X>, <Ctrl-C> and <Ctrl-V> assignments. If you don't know them (we never could remember them!), there is no reason to learn them because there are easier ways to use the scratchpad.

HINTS: The scratchpad functions can be accessed from the toolbar. For cut and paste operations within VEDIT, it is better to use the scratchpad instead of the Windows clipboard.

It is also easy to use the scratchpad by double-pressing the hot-keys for the text register functions. These are normally the **<Numpad+>**, **<Numpad->** and **<Numpad*>** on the numeric pad.

Double-press **<Numpad+> or <Numpad->**, when no block is highlighted to copy/move the current line to the scratchpad. Then double-press **<Numpad*>** to insert the line somewhere else. This is a quick way to copy/move a single line.

When a block is copied to the scratchpad (or text registers), VEDIT remembers what type of block (stream, column or line) it is. When the scratchpad is inserted into the text, it is inserted in the manner appropriate for that type of block. For example, a line block is inserted at the beginning of the current line. This is the primary advantage of using the scratchpad instead of the Windows clipboard.

Accessing Other Text Registers

To copy a block of text into a text register, highlight the block of text and select **{BLOCK, Copy to register}** (<**Numpad**+>). Alternatively, to move (or "cut") the block to a text register, select **{BLOCK, Move to register}** (<**Numpad**->).

VEDIT then prompts for the register's number. Enter the desired number "0" through "100"; "0" is the default.

HINT: When prompted for a text register number, you can easily select the default "scratchpad" register "0" by pressing any function/control key. For example, with the normal keyboard layout, to copy to register "0", simply press **<Numpad+>** twice.

DOS: ***UNIX:*** ***QNX:*** The non-windows versions have an optional "terse" method of selecting register numbers. To select from the first ten registers simply press "0" through "9"; you don't need to press **<Enter>**. To select other registers, first type "." (period), type the number and then press **<Enter>**. For example, to select register 20, enter "**.20**". This is convenient, because in practice, you rarely use more than the first ten registers.

The text register selection dialog box gives you the option of appending the block to the existing contents of a text register or inserting the block at the beginning of the existing contents.

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}** (<**Numpad***>). The register contents can optionally overwrite the existing text.

Block options - Fill and Overstrike

When a block of text is moved (cut) to a text register with **{BLOCK, Move to register}**, the original block is normally deleted. Optionally, the block can be filled with spaces or a configurable character. In the "Move to register" dialog box, select the "**[ ] Fill buffer text**" option.

Both character and columnar blocks can be filled. The fill character can be changed with **{CONFIG, Tab/Fill, Block fill character}**.

NOTE: An option in **{CONFIG, Emulation, Expand <Tab> with spaces}** fills a columnar block with the optimal number of Tab characters and spaces. The default is to fill only with spaces.

When a block of text is inserted (pasted) from a text register with **{BLOCK, Insert register}**, you can optionally overwrite the existing text. In the "Insert register" dialog box, select the "**[ ] Overwrite buffer text**" option.

Since block operations using the "Fill" and "Overwrite" options do not change the size of the file, these options are automatically selected when editing in *Overwrite-only* mode. This mode is selected with **{CONFIG, File handling, Overwrite-only mode}**. To prevent file corruption, binary and database file editing should normally be done in overwrite-only mode.

In Overwrite-only mode, **{BLOCK, Copy / Move to cursor}** also overwrite the existing text at the cursor. **{BLOCK, Move to cursor}** also fills the original block.

NOTE: A highlighted block can simply be filled (e.g. with spaces) by selecting **{BLOCK, Edit/Translate, Block fill}**.

Emptying a Text Register

If you receive the error message "NOT ENOUGH MEMORY FOR OPERATION" when attempting to copy/move a block to a text register, you may be able to perform the operation successfully if you first empty any text registers that are no longer needed.

Unless you receive this error message, it is usually not worth while to empty text registers.

➢ **To empty text registers that are no longer needed:**

1. Use **{HELP, Text registers}** to see how much is stored in each register and the first few bytes of their contents. Note which register(s) you want to empty; only worry about registers that have 1000 or more bytes in them.
2. Select **{BLOCK, Set marker}** and then immediately **{BLOCK, Copy to register}**. It does not matter where the cursor is. Select the register to empty.

   Repeat this step for any other registers to be emptied.

Text Register Usage

The 100+ text registers serve two primary purposes:

- For "cut and paste" operations, where they temporarily hold a block of text.
- To hold sequences of commands in the VEDIT PLUS macro language which may be executed as "command macros".

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

With over 100 text registers available, it is easy to forget what each register contains. Several text registers are also reserved for special purposes. We recommend the following organizational scheme for using registers:

0 This is the default "scratchpad" register.

1 - 9 These are used as additional "cut and paste" registers from Visual Mode.

10 - 99 These are used to hold command macros or as string variables in command macros. However, they can also be used for "cut and paste" operations.

100 This register is used by the `startup.vdm` file and any auto-execution macros. It is also the default register for **{MISC, Load and execute macro}**. It should be reserved for the "main" macro that is running.

101 - 127 These registers are reserved for use by the VEDIT PLUS macro language and are described below. They cannot be accessed from Visual Mode.

Usage (Technical)

101 Should be reserved for the "subroutine" macros used by the main macro executing in register 100.

102 Should be reserved for the "locked-in" macro used by the main macro executing in register 100.

103 - 106 Temporary registers used as needed by keystroke macros. This prevents keystroke macros from interfering with command macros that may be running.

107 - 109 Reserved for use by the "File open/close macros", "Buffer switch macro" and the "Template editing macro". This prevents these special event macros from interfering with other macros that may be running.

110 The "File open event macro". It is executed immediately after each file is opened, when **{CONFIG, Programming, File-type specific configuration}** or the equivalent **CONFIG(PG_F_AUTO_CFG)** is enabled.

111 The "File close event macro". It is executed just before each file is closed.

112 The "File pre-open event macro". It is executed just before each file is opened.

113 The "File post-close event macro". It is executed immediately after each file is closed.

114 The "Buffer switch event macro". It is executed immediately after each buffer switch in Visual Mode or due to the macro language command **Buf_Switch(***r*,EVENT).

115 The "Template editing macro". It is executed for each character entered in Visual Mode when **{CONFIG, Programming, En-**

able template editing} is enabled. It can be loaded with **{MISC, Load template file}**.

116 - 117 Reserved for use by future event macros.

118 Internally used text register. It is used by and emptied by many block commands.

119 Should be reserved for "subroutine" macros set up within a ".key" file.

120 Internally used text register. It is emptied with each keystroke and by many block commands.

121 Internally used register that holds the filename from the File selection dialog boxes.

122 Internally used to load the PRINT.VDM, SALLBUFF.VDM, SRCHINCR.VDM, LOADSYN.VDM, KEYEDIT.VDM and VEDITSAV.VDM macros.

123 This register holds the custom editing functions for the **{TOOL}** menu. Otherwise it must be empty.

124 This register holds the custom editing functions for the **{USER}** menu. Otherwise it must be empty.

125 Internally used to hold the keyboard layout in a binary format. It MUST NOT be altered. (125 - 127 are accessible for use by the VEDITSAV.VDM macro which saves the entire VEDIT environment.)

126 Internally used to hold the current window structure. It MUST NOT be altered.

127 Internally used to hold the last command line entered at the "COMMAND:" prompt. (It has a constant size.) It MUST NOT be altered.

To protect users from unintentionally overwriting text registers, the Visual Mode can only access registers 0 through 100. The **Reg_Prot( )** command can also be used by a command macro to write-protect the text registers it uses.

More Text Register Notes

The description for **{BLOCK, Copy to register}** in Chapter 6 (Menu Reference) gives step by step examples for using the text registers.

All text register contents are lost when you exit VEDIT unless you are using the "Edit Session Restore" feature. When an edit session is restored, all text registers are restored.

Command macros have the ability to "write-protect" text registers used by the macro so that you do not inadvertently alter them.

Cut & Paste Huge Blocks

Many "Cut & paste" operations are currently limited to a maximum block size of about 250,000 bytes for the Windows 32-bit version and 60,000 bytes for other versions. However, by writing a block out to disk with **{BLOCK, Write to disk}** and then inserting it with **{EDIT, Insert file}** you can cut & paste huge multi-megabyte blocks.

This limitation occurs because the text registers are currently limited in how much they can hold. The "scratchpad" is one text register. The functions **{BLOCK, Copy/move to cursor}** use an internal text register. If you have other blocks or macros stored in the text registers, the maximum cut & paste block size may be much smaller than 250,000. If the block is too large to fit into a text register, you will receive the error message "BLOCK TOO LARGE FOR TEXT REGISTER".

- **You can cut & paste blocks of any size:**

1. Highlight the block and select **{BLOCK, Write to disk}**. Both non-columnar (stream and line) and columnar blocks can be written to disk.

   Choose a filename such as "temp".

2. Move to the destination for the block. It can be in the same file or a different file.

3. Select **{EDIT, Insert file}**. Enter the same filename, e.g. "temp".

   If you wrote a columnar block to disk in step 1., be sure to select "**[ ] Columnar (block) insert**". This will insert the file as a columnar block in the same way that **{BLOCK, Insert register}** does.

(The 32-bit Windows version supports multi-megabyte clipboard operations, and 250,000 bytes in the text registers.)

The Windows Clipboard

Blocks of text can also be copied/moved to the Windows clipboard and inserted from the clipboard. It is similar to VEDIT's scratchpad. Although the clipboard is the only method for "cut and paste" with most other programs, we highly suggest using VEDIT's scratchpad and other text registers for all cut and paste operations within VEDIT.

The clipboard should only be used for exchanging text with other Windows programs. The clipboard does not handle columnar blocks as well as the text registers, and does not support binary data.

You can cut or copy the currently marked (highlighted) block to the clipboard, or paste the clipboard into the file being edited. You can paste the clipboard as either a "stream" or "columnar" block. Selecting "Paste clipboard" inserts the entire clipboard at the cursor position. Selecting "Paste columnar clipboard" inserts each line of the clipboard into successive lines of your file, each time starting at the current column.

See Also:

{EDIT, Clipboard} functions in Chapter 6 (Menu Reference).

Block Indenting

You can easily change the indentation of an entire block of text. This is especially useful when editing structured programming languages such as C and Pascal. When the cursor is within a highlighted block of text, **{EDIT, Indent}** and **{EDIT, Undent}** indent/undent all lines in the block by the current "indent increment", typically 4 columns. The indent increment can be changed with **{CONFIG, Programming, Indent increment}**.

HINT: With the default keyboard layout and configuration, the keys **<Tab>** and **<Shift-Tab>** are equivalent to **{EDIT, Indent}** and **{EDIT, Undent}** when the cursor is within a highlighted block.

➢ **To change the indentation of an entire block of text:**

1. Highlight the lines that need re-indenting. Be sure the cursor is within the highlighted block or immediately after it.
2. Press **{EDIT, Indent}** (<**F8**>) and **{EDIT, Undent}** (<**F7**>) until the block is correctly indented.

-OR-

Press <**Tab**> and <**Shift-Tab**> to correctly indent the block.

3. Use any desired method to remove the block highlighting. E.g. press **[CANCEL]** (<**Ctrl-**\>).

Notes:

The file KEY-MAC.LIB contains several keystroke macros that can also perform block indenting. One aligns the left edge of the current line with the cursor and advances to the next line. Another aligns the current line with the previous line and advances to the next line.

See Also:

{EDIT, Indent} in Chapter 6 (Menu Reference).

Columnar Blocks

VEDIT can manipulate columns of text. A *columnar block* is a rectangle of characters in your file. 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. All of VEDIT's block operations work with columnar blocks. You can also restrict a search/replace operation to the columnar block.

Columnar blocks are especially convenient for editing tabular data such as a spreadsheet or a database. For example, you could copy a table of numbers, say between lines 10 and 20 and between columns 30 and 40 to a text register. This columnar block of numbers can then be inserted anywhere else.

To make columnar block operations appear as natural as possible, VEDIT performs some additional manipulations on your text.

- It ensures that the columnar block being copied has a flush right margin. If any lines being copied are shorter than the block's right margin, they are padded with spaces to make them flush. This ensures 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.
- Any tab characters in the columnar block being copied are converted to spaces. (This is necessary for columnar operations to work as expected.) When the columnar block is inserted, you can select with **{CONFIG, Tab/Fill, Expand <Tab> with spaces}** whether these spaces (and adjacent spaces in the existing text) are converted back to Tab characters.

In some cases, such as inserting a columnar block at the end of text lines, these extra padded spaces become trailing spaces. These extra spaces are trimmed by default, but this can be changed with **{CONFIG, Tab/Fill, Trim spaces after columnar operation}**.

NOTE: 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 when working with columnar blocks.

Columnar Block Examples

To perform columnar block operations mark the desired block with **{BLOCK, Set column marker}** (**<Alt-I>** or toolbar). 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.

NOTE: 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.

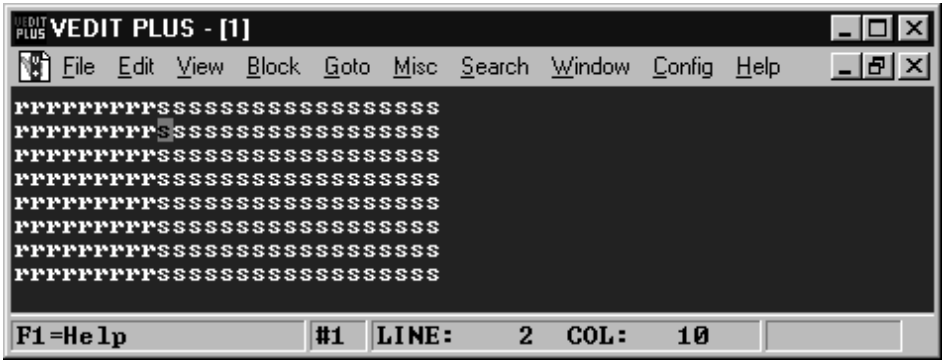
The following screen shows a columnar block highlighted. Note that the cursor is in the lower right corner of the block.

After copying this block to a text register, the register will contain:

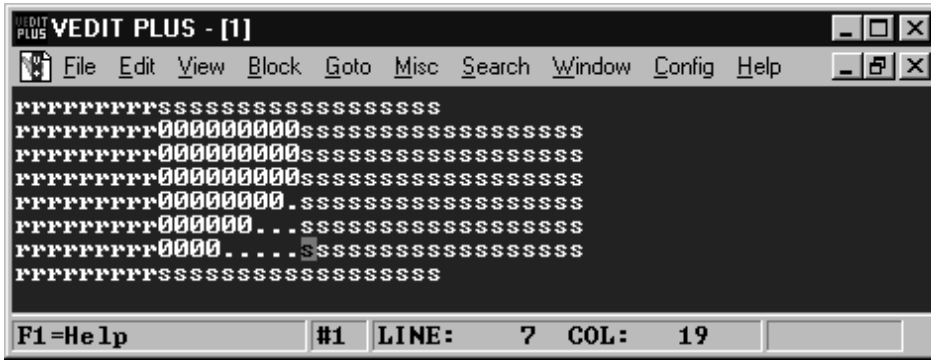
```
000000000
000000000
000000000
00000000.
000000...
0000.....
```

The "." are padding spaces that were added to the register in order to give it a flush right margin.

Now consider the following text before inserting this register.



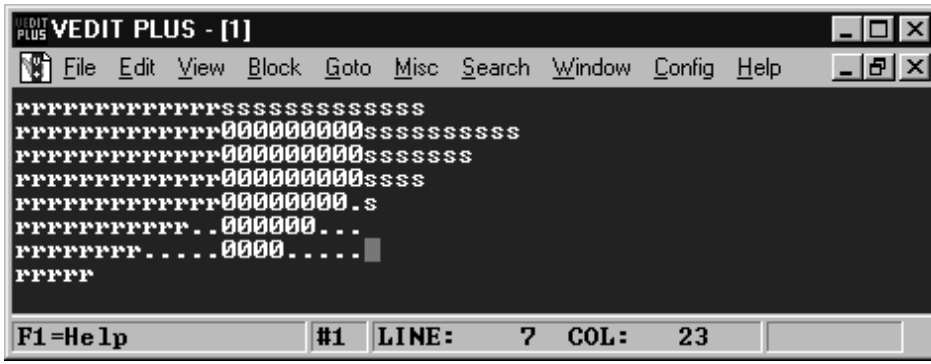
After inserting the register, the screen will display:



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.

After inserting the same register, the screen will display:



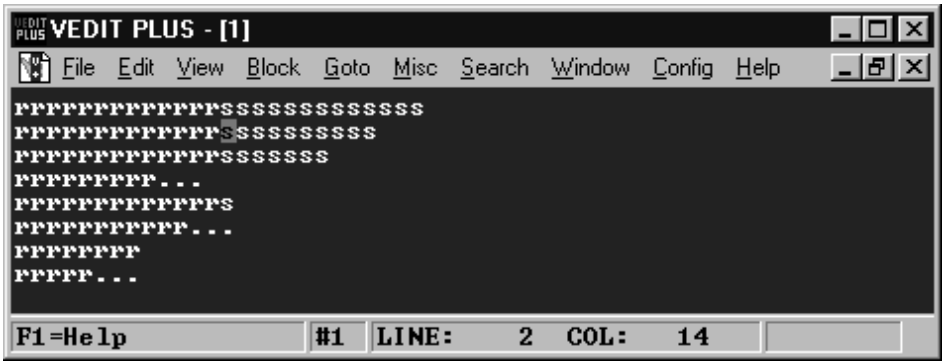
By padding text lines 6 and 7 with spaces, the inserted block also remains aligned.

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, Tab/Fill, Trim spaces after columnar operation}** 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.....
```

Consider the following text which contains three trailing spaces on lines 4, 6 and 8. (Remember that trailing spaces are invisible unless you change **{CONFIG, Characters/Cursors, Newline display character}** to display the location of the "newline" at the end of each line.)



With **{CONFIG, Tab/Fill, Trim spaces after columnar operation}** enabled, inserting the text register will change the screen to:

The trailing spaces, including the existing ones on lines 4, have been trimmed. However, the trailing spaces on line 8, which was not involved in the insertion, still remain.

As explained earlier, Tab characters in a columnar block operation are first converted to spaces and can optionally be converted back to Tab characters. However, a single space is never converted to a Tab character. Also, the first two spaces following a "." (period), "!" and ":" are not converted to a Tab character. This makes columnar block operations more compatible with the needs of word processing.

The above discussion also applies to columnar block copy/move performed with **{BLOCK, Copy to cursor}** and **{BLOCK, Move to cursor}**. These are implemented with an internal text register and therefore operate identically.

Notes:

Use **{BLOCK, Edit/translate, Detab} to convert Tab characters in a file to spaces.**

Use **{USER, Remove trailing spaces}** to remove all trailing spaces from a file.

Translating a Block or File

A marked block or entire file can be translated using the built-in EBCDIC table, the supplied ANSI table or with a user-created table. When a block/file is translated, each byte is simply converted to another byte according to the current table; the size of the file does not change.

Translating between EBCDIC and ASCII

For example, to translate an EBCDIC file downloaded from an IBM mainframe into ASCII for use on a PC:

➢ **Translate a file from EBCDIC to ASCII:**

1. Open the EBCDIC file in the normal manner, e.g. with **{FILE, Open}**.
2. Select **{BLOCK, Select all}** to mark the entire file as a block.
3. Select **{BLOCK, Edit/Translate, Translate from EBCDIC}** to translate the file to ASCII.

If the original EBCDIC file consisted of fixed-length records without end-of-record characters, you may want to add an ASCII "newline" (Carriage-Return and Line-Feed) to the end of each record so that Windows/DOS programs can more easily read it.

4. Select **{CONFIG, File handling, File type}** and set the correct record length. Records should now be nicely aligned on the screen.
5. Select **{BLOCK, Convert newlines, Convert macro}**. In the macro's menu, select "Fixed length records to Newlines".

   You may now want to change **{CONFIG, File handling, File type}** to "0" or "1" to recognize ASCII "newlines" and make the file more readable on the screen. This does not affect the translated file itself.
6. Select **{FILE, Close buffer}** or **{FILE, Exit}** to save the translated file.

Instead of translating an EBCDIC file, you can also display the file in ASCII (instead of gibberish) by selecting **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar) several times to enter "EBCDIC" mode. In this mode, the same translation table is used, but only for display purposes; the EBCDIC file itself is not changed.

Similarly, an ASCII file can be translated to EBCDIC with **{BLOCK, Edit/Translate, Translate to EBCDIC}**.

IMPORTANT: Since IBM PC ASCII and EBCDIC have somewhat different character sets, not all characters will translate without problems. In particular, there is no equivalent of most IBM PC graphics characters. Therefore, if you translate a file to EBCDIC and then back again to ASCII, you may not have the same file again. Some punctuation and most control and graphics characters will have changed.

The EBCDIC translate table EBCDIC.TBL is built into VEDIT and does not need to be loaded. However, for custom needs you can modify the EBCDIC.TBL file and then load the revised EBCDIC table into VEDIT.

NOTE: Conversion packages are available for translating EBCDIC files with packed (signed) decimal, packed binary, zoned and other special (COBOL) fields. Please see the "EBCDIC" page on our Web site for details.

Translating between ANSI and ASCII

ANSI.TBL is supplied for translating between the IBM PC graphics characters and the ANSI (Windows) character set. This is particularly useful for translating foreign characters from their IBM PC value to the value needed for most Windows word processors. For example, the "ü" (small umlaut u) has an IBM PC value of "129", but an ANSI value of "252". ANSI.TBL does not change any normal characters with values of less than 128. As with most translations, some characters cause problems because there is no equivalent. For example, there is no exact equivalent to the IBM PC box-drawing characters.

- **To load the ANSI.TBL translation table:**

  1. Select **{BLOCK, Edit/Translate, Load translate table}**. At the filename prompt enter "ansi.tbl".

After ANSI.TBL is loaded, note that the **{BLOCK, Edit/Translate}** menu now displays "Translate from ANSI" and "Translate to ANSI" to reflect the new translation table's name. Similarly, when changing display modes with **{VIEW, Toggle display mode}**, the status line will read "ANSI" instead of "EBCDIC".

Creating Your Own Translation Table

You can easily create a translation table for your own needs. A 521 byte translation file consists of two 256 byte translation tables (a "to" table and a "from" table) followed by an eight character name and a Null (00) byte.

The easiest way to create your own translation table is by modifying the supplied file USER.TBL.

See the on-line help topic "Translating a Block or File" for details. (DOS: On-line help topic "TRANSLATE".)

Sorting Lines in a Block / File

VEDIT has both a sort function **{BLOCK, Edit/translate, Sort lines}** for sorting lines (single line records), and a macro SORT.VDM for sorting records consisting of multiple lines. Both support variable-length and fixed-length records.

The **{BLOCK, Edit/translate, Sort lines}** function is described here. The SORT.VDM macro is described in Chapter 5 (Advanced Topics).

{BLOCK, Edit/translate, Sort lines} sorts the selected lines after you highlight the field to be sorted as a columnar block. This field is the "sort key"; for example, it could be the Last-name or Zipcode in a database.

- **To sort all lines in a block or entire file:**

1. Position the cursor on the first line to be sorted and in the left-most column of the field to be used as the "sort key".

   Select **{BLOCK, Set column marker}** (**<Alt-I>** or toolbar) to begin a columnar block.

2. Position the cursor on the last line to be sorted and in the right-most column of the field to be used as the "sort key". To sort an entire file, this must be the last line of the file.

   Select **{BLOCK, Set column marker}** (**<Alt-I>** or toolbar) again.

3. Select **{BLOCK, Edit/translate, Sort lines}**. The lines will be sorted and VEDIT will display its progress. A 250 Kbyte file with 1000 lines takes about one minute to sort.

NOTE: VEDIT's sorting may not be fast enough for multi-megabyte files. It is only intended as a convenient way of sorting smaller files up to a few megabytes in size. A dedicated sort program or a database program may be needed to sort very large files. A much faster sort is planned for late 1999.

Sorting by multiple fields (major and minor keys)

When VEDIT's sorting encounters multiple lines (records) with the same "key", it leaves them in the same order as they are encountered in the file. Therefore, to sort by a major key and, within the major key, by a minor key, first sort the entire file using the minor key field. Then sort the entire file using the major key field. The file will then be sorted by both keys.

Printing in VEDIT

Printing is very flexible in VEDIT. You can print an entire file or just a highlighted block. Since you have complete control over *print jobs*, you can even print several blocks on the same page.

Configurable "margins" are typically used to prevent the text from printing on the extreme edges of the paper. Text can be printed single, double or triple spaced. Or you can print a file in "raw" mode — exactly as-is, without adding any margins.

Advanced options include setting any desired "print mode", which determines how control and graphics characters are printed. A file can be printed in hexadecimal; an EBCDIC file can be printed in ASCII.

Windows Version:

The font used for printing can be selected with **{CONFIG, Printer, Printer font}** or in the print dialog box. All characters are printed in the same font and size.

VEDIT can open a file that was created with "Print to file" in a program that supports fonts, such as Microsoft Word (tm). If this file is then printed in "raw" mode, it will print correctly since all of the font selection information is embedded within the file.

The optional "Print job start/finish strings" are of limited use since Windows sends its own strings with each print job.

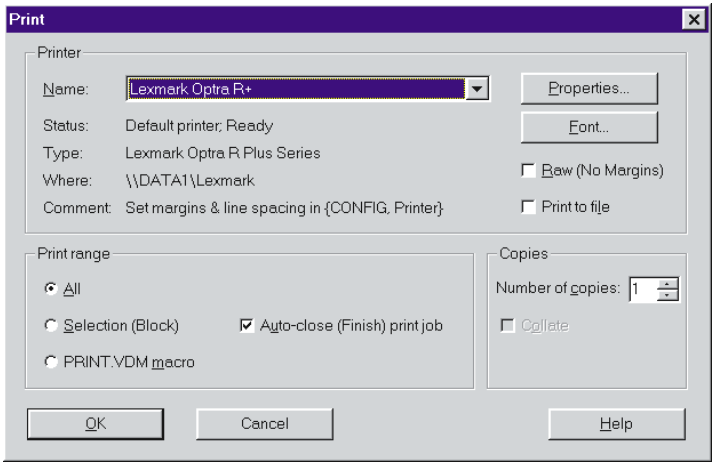
DOS Version:

VEDIT can print to a local or network printer on any desired parallel or serial port, to the default DOS "PRN" port, or to a file. When printing to a file, you can always print to the same filename or be prompted for the filename each time.

An optional "Print job start string" (initialization string) can be sent to the printer to select any desired font, pitch, font size and weight (e.g. bold). This is very technical and requires knowledge of the control sequences needed by your specific printer. After the text is printed, an optional "Print job finish string" can be sent to reset the printer.

See Also:

On-line help for the **{FILE, Print}** dialog box.



Basic Operation

- **To print the entire file:**
  1. Select **{FILE, Print}** (default: **<Ctrl-P>**).
  2. Assuming a block of text is not highlighted, "**All**" is automatically selected; otherwise manually select "**All**".
  3. Select the **[Ok]** button or press **<Enter>**.
- **To print a single block of text:**
  1. Highlight the desired text as a block.
  2. Select **{FILE, Print}** (default: **<Ctrl-P>**).
  3. If the cursor is within the highlighted block, "**Selection (Block)**" is automatically selected; otherwise manually select it.

     Leave "**Auto-close (Finish) print job**" enabled.
  4. Select the **[Ok]** button or press **<Enter>**.
- **To print multiple blocks of text (e.g. on the same page):**
  1. Highlight the desired text as a block.
  2. Select **{FILE, Print}** (default: **<Ctrl-P>**).
  3. If the cursor is within the highlighted block, "**Selection (Block)**" is automatically selected; otherwise manually select it.
  4. Disable "**Auto-close (Finish) print job**".
  5. Select the **[Ok]** button or press **<Enter>**.
  6. Repeat steps 1. and 2. to highlight and print the additional blocks of text. They will be printed one after another. Since the print-job has not yet been closed, a different dialog box is displayed.

7. Select **{FILE, Print}** again; then select the **[Finish/Eject]** button to finish (close) the print-job and release it to the printer.

Printer Margins

All printing margins can be selected from **{CONFIG, Printer}**.

A "Top margin" and "Bottom margin" are used to prevent the text from printing at the very top and bottom of a page. Assuming a "Paper length" of 60 lines with a 3 line margin at the top and bottom of each page, 54 lines of text are printed on each page.

In the Windows version, **{CONFIG, Printer, Paper length}** should normally be set to "0=Auto". The number of lines printed per page is then automatically adjusted according to the font size, the paper size and its orientation (portrait or landscape).

In the DOS version, the "Paper length" must be set to agree with your particular printer. Dot matrix printers typically use 66 lines per page (6 lines per inch and 11 inch paper results in 66 lines). Since most laser and ink-jet printer cannot print on the top and bottom 1/2 inch of the paper, they typically have a default of 58 to 62 lines per page. However most laser printers can be set to 66 lines per page — they then print slightly more than 6 lines per inch.

TROUBLE-SHOOTING: If every other page is printed with just a few lines of text, this is usually caused by VEDIT's "Paper length" being set too big for your printer. Try a smaller value.

A "Left margin" is used to prevent the text from printing at the very left edge of the paper. The default value of 5 columns typically gives a one inch margin on a laser printer.

NOTE: Don't indent all text in your file for printing purposes; instead, use **{CONFIG, Printer, Left margin}** to position your text on the printed paper.

An optional "Right margin" wraps very long text lines onto multiple printed lines; otherwise a printer typically truncates long lines. By default, the "Right margin" is disabled so that each line in your file prints as one line, even though it may be printed truncated.

Using the PRINT.VDM Macro

Selecting "PRINT.VDM macro" in the **{FILE, Print}** dialog box causes the entire file, or just the highlighted block, to be printed with the filename, date and page number at the top of each page. It is ideal for printing source code modules and other text files.

The PRINT.VDM macro uses the configured "Paper length" and printer "Left margin", but always uses a top and bottom margin of two lines.

This print function automatically loads and executes the the macro **`print.vdm`**. This macro is not overly complex and simple changes can be

made to it without having to fully understand the VEDIT PLUS macro language.

`print.vdm` documents how to optionally print a ruler at the top and bottom or each page, or line numbers and/or file positions on the left side of the page.

VEDIT PLUS: The operation of **`print.vdm`** can easily be modified by anyone familiar with the VEDIT PLUS macro language. **`print.vdm`** is intended as a macro example which is relatively easy to understand and expand.

Much more sophisticated custom formatters can be written in the macro language. If they are named **`print.vdm`**, they can be accessed from the print dialog box.

See Also:

The topic "PRINT - Print Macro" in Chapter 5.

Print Display Mode

Similar to the screen "display mode", you can determine how control and graphics characters are printed. You can also print in hexadecimal, octal or print an EBCDIC file.

Since printers respond to "control sequences" to changes fonts, turn on underlining and select other features, you have to be careful when printing files that contain control characters. If these control characters are intended for controlling the printer, they should be printed as-is. Otherwise, they should be converted to the "^***x***" format.

{CONFIG, Printer, Print mode} determines the printing mode. The most common values are:

0 (Default) Print in the same mode as the current display mode, i.e. what-you-see-is-what-you-get. Depending upon the display mode, control characters may be sent as-is to the printer or can be expanded to the "^***x***" format.

2 Only expand tab characters with spaces. All other control characters are sent as-is to the printer. This lets you embed printer control sequences in your text to control fonts and other features.

274 Safe mode. All control characters are expanded to the "^***x***" format.

1024 Print all control characters, including tab characters, as-is. This mode is automatically used when "Raw" is selected in the Print dialog box.

The on-line help for **{CONFIG, Printer, Print mode}** lists all possible values.

EBCDIC and other Translate Tables

(This is an advanced topic.)

With a print mode of "8192", VEDIT translates each character that is sent to the printer. The EBCDIC translate table is built into VEDIT and, therefore,

you can load an EBCDIC (IBM mainframe) file and print it on your ASCII printer. (All printers connected to a PC are ASCII.)

Note that this does not modify or translate the file itself, only the characters sent to the printer are translated.

VEDIT can load custom translation tables and this can be very useful for many printing applications. Here are some examples:

- You might want to print a file with some or all control characters translated to a period ".".
- Print a file with the high (8th) bit stripped from all graphics characters.
- Only translate one or two characters that are causing trouble, for example translate the letter "O" to the digit "0".

Any of these tasks can be done with a custom translation table. Here is an overview of the steps involved.

1. Create the desired custom translation table. See the topic "Translating a Block or File" earlier in this chapter. It may take you about an hour the first time; 15 minutes when you are familiar with the process.
2. Load the custom translation table with **{BLOCK, Edit/translate, Load translate table}**.
3. Select **{CONFIG, Printer, Print mode}** and enter a value of "8192".
4. Select **{FILE, Print}** to print the file.

Print "Jobs" and [Finish/Eject]

(This is a moderately technical topic.)

VEDIT gives complete control over *print jobs* so that you can print several blocks of text on the same page. (Most other programs only let you print one block per page.) You will typically only notice the details of print jobs when printing blocks.

When VEDIT begins printing, it performs the following steps:

1. Opens a new "print job", which is similar to opening a file. (If you are printing to a file, it really does open a file.)
2. Optionally sends the configured "Print job start string" to the printer.
3. Sends the desired text to the printer. It will be formatted according to the current margins unless "Raw" mode is selected.

When printing with "All - entire file", VEDIT automatically finishes and closes the print-job. However with "Block only" you must explicitly select **[Finish/Eject]** after you have printed the last block.

When VEDIT finishes printing, it performs the following steps:

1. Sends a "page eject" to the printer, assuming that **{CONFIG, Printer, Page eject on Finish/Eject}** is set. It is set by default, but can be disabled in case your network printing also adds a page eject to each print-job.

Depending upon **{CONFIG, Printer, Enable Form-Feed}**, either a single Form-Feed character or multiple Line-Feeds are sent to perform the page eject. The default is to send a Form-Feed, which performs a page eject on (nearly) all printers.

2. Optionally sends the configured "Print job finish string" to the printer.
3. Closes the print-job. This releases the print-job and it should begin printing. (When printing to a file, it closes the file.)

Nothing will print until VEDIT closes the print-job. Similarly, when printing to a file, the file will be empty until you close the print-job. (When you exit VEDIT, any open print-job is automatically closed).

(DOS version: Text may print before VEDIT closes the print-job because characters are sent directly to a local printer.)

Print Job Start/Finish Strings

(This is a technical topic.)

Windows: Since Windows sends its own print-job start/finish strings for most printers, attempting to set them within VEDIT will have no effect.

The only exception is the "Generic text" printer. If you select this printer, Windows does not send its own print-job start/finish strings and you can send the strings from VEDIT. You can easily add the "Generic text" printer from within the Windows 95/98/NT Printer Control Panel.

An optional "Print job start string" can be sent at the beginning of each print-job to initialize the printer. This is typically used to select a font, pitch, size or weight. This is particularly useful on network printers where different users might have different printing preferences.

Similarly, an optional "Print job finish string" can be sent at the end of each print-job. This typically resets the printer to its default state so that the next program used is not affected by VEDIT's printing. The finish-string is not as useful in the Windows version, because Windows sends it own finish-string at the end of each print job.

The print-job start and finish strings can be up to 32 characters in length. However, by specifying a string of "@filename", an arbitrarily long string can be sent to the printer. VEDIT searches for the file first in the current directory and then in the *VEDIT Home Directory*.

These example start (init) strings select font features on an HP or other PCL language compatible printer:

Example Print-job Start Strings

Function	Command	Decimal Value
10.0 Pitch	<Esc>(s10H	27 40 115 49 48 72
12.0 Pitch	<Esc>(s12H	27 40 115 49 50 72
Bold	<Esc>(s3B	27 40 115 51 66
Normal	<Esc>(s0B	27 40 115 48 66
Reset printer	<Esc>E	27 69

Because the print-job start/finish strings are not used very often in the Windows version, they can only be set in the **`vedit.cfg`** file. In the DOS version, they can be changed in the **{CONFIG, Printer}** sub-menu.

➢ **To enable (or change) print-job start/finish strings:**

1. Edit the **`vedit.cfg`** file as described in Chapter 8 (Configuration).
2. Change **Config(P_E_STRING)** to a value of "3".
3. Near the end of the file, find the items **Config_String(PR_START, "")** and **Config_String(PR_FINISH,"")**.

   Enter the desired start/finish strings between the double quotes. To enter control characters such as **<Esc>**, precede them with **[ENTER CTRL]** (default: <**Ctrl-Q**>).

   For example, to enter "**<esc> (s12H**" from the above example, place the cursor between the quotes and type the following keys:

   <Ctrl-Q> <Esc> (s 1 2 H

Another way to enter the print-job start/finish strings is to hold down the <**Alt**> key and then type each decimal value on the numeric keypad.

Search and Replace

This topic describes the powerful search and replace capabilities in VEDIT called "*pattern matching*" and "*regular expressions*".

Searches normally use *pattern matching*. This is a powerful searching syntax, unique to VEDIT, that has "wildcard" characters for matching letters, digits and much more. It is easy to use; only the character "|" has a special meaning.

Searches can alternatively use *regular expressions*. This is a powerful search (and replace) syntax from UNIX. It is much more complex and most punctuation characters have a special meaning. It is also slower than pattern matching.

You can also search in *Simple* mode without any pattern matching or regular expressions. It is only recommended for very novice users because no "wildcard" searching is possible. (It is not faster than pattern matching.)

The desired search mode can be selected in the Search dialog box. The default mode is selected with **{CONFIG, Search options, Default search mode}**.

Pattern Matching

Pattern matching can search for types of characters such as "any digit", or for characters that meet special conditions such as "occurring at the beginning of a line".

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.

NOTES: "**|**" is the "pipe" character, which is <Shift>-**** on the keyboard. All pattern matching codes begin with this character.

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

Most of the pattern matching codes only have a special meaning in the search string; they have no meaning in the replacement string. If you need "variable" characters in the replacement string, you must use regular expressions.

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

|D|D	Search for two consecutive digits.
|!|D|D|D|!|D	Search for next two digit number. (It will not match a three digit number.)
|<note	Search for a line beginning with the word "note".
|W|>	Search for "whitespace" (any number of spaces and tabs) at the end of a line.
t|A|A|An	Search for any five letter word beginning in "t" and ending in "n".

|000 Search for the "Null" character (value 000).

The pattern matching codes are:

|A Match any alphabetic letter, upper or lower case. It supports non-english letters, such as "umlauts", if **{CONFIG, Search options, Support non-english characters}** is enabled.

|B Match a blank - a single Space or Tab. See also "|**W**" and "|**X**".

|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 graphics character - characters with decimal value greater than 128. (Useful for finding stray graphics (8-bit) characters in a file.) See also "|**K**".

|H*hh* Match the character with hexadecimal value '*hh*'. Both digits ***MUST*** be present. This code can also be used in the replacement string.

|I Match any word separator - Space, Tab, any control character, or one of the additional configurable word separators defined by **Config_String(WORD_SEP)**.

|K Match any (non-standard) control character other than Tab, Carriage-Return and Line-Feed. (Useful for finding stray control characters in a file.) See also "|**G**".

|L Match the "newline" character(s) Carriage-Return and/or Line-Feed depending upon the file type. With Windows/DOS files, the Carriage-Return is optional. Similar to "|**N**".

|M Match multiple characters - zero, one or more characters until the string following the "|**M**" is satisfied. This code is not meaningful as the first item in a search string. See also "|**Y**" and the following sub-topic "Matching Multiple Characters".

|N Match the "newline" Carriage-Return and/or Line-Feed depending upon the file type. With Windows/DOS files, the Carriage-Return is mandatory. This code can also be used in the replacement string. See also "|**L**".

|O*ooo* Match the character with octal value '*ooo*'. Three digits MUST be present. This code can also be used in the replacement string.

|P Match any parenthesis - { }, [ ], < > and ( ). (Internally used by **{GOTO, Matching ()}**.)

|S Match any separator - a character which is not a letter, a digit or underscore "_". Space, Tab and all control characters are separators. Graphics characters (value 128-255) are not separators.

|T Match the ASCII Tab character (hex 09). This code can also be used in the replacement string.

|U	Match any upper case letter. This pattern supports non-english letters, such as "umlauts", if **{CONFIG, Search options, Support non-english characters}** is enabled.
|V	Match any lower case letter. See description for "**|U**".
|W	Match "white space" - one or more Spaces and/or Tabs. See also "**|B**" and "**|X**".
|X	Match extended white space - one or more Spaces, Tabs, Carriage-Returns and/or Line-Feeds. See also "**|B**" and "**|W**".
|Y	Match zero, one or more characters until the immediately following character or pattern matching code is satisfied. This code is not meaningful as the first item in a search string. See also "**|M**".
|*ddd*	Match the character with decimal value '*ddd*'. This code can also be used in the replacement string.
|000	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". This code is not meaningful as the first item in a search string.
|<	Match the beginning of line position — the following matched characters must occur at the beginning of a line. (See below.)
|>	Match the end of line position — the preceding matched characters must occur at the end of a line. (See below.)
|@(*r*)	Use the contents of text register '*r*' in this position in the search string. This code can also be used in the replacement string.
|?	Match any single character; this is the simple "*wildcard*" similar to "?" in filenames.
|!	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|!s". Think of "|!x" as "not x".
||	Match the "|" character. You need a double "||" to search for a single "|" in your text. A double "||" is also needed on the replacement side.

Matching the "Newline" - Carriage-Return and Line-Feed - with "|L" and "|N"

The pattern matching codes "**|L**" and "**|N**" match the newline character(s) Carriage-Return and/or Line-Feed depending upon the current file type.

When **{CONFIG, File handling, File type}** is set to "0" (Windows/DOS text file), they match a Carriage-Return and Line-Feed pair as the "newline" characters. When set to "2" (Mac text file) they match a single Carriage-Return character. Otherwise, they match a single Line-Feed character.

"|**L**" and "|**N**" are similar, but not identical. For file type "0" (Windows/DOS text), "|**L**" treats a Carriage-Return as optional, while "|**N**" requires both a Carriage-Return and Line-Feed. Therefore, "|**L**" is preferred in search strings because it handles Windows/DOS text files that might be missing some Carriage-Returns. For other file types, "|**L**" and "|**N**" are identical.

"|**N**" is equivalent to "|**013**|**010**", "|**010**" or "|**013**", depending upon the file type. "|**N**" works slightly faster than "|**L**", and "|**N**" can be used in the replacement string. Using "|**N**" is a convenient way to enter a multiple-line replacement string. In summary:

- "|**L**" is preferred in search strings.
- "|**N**" can be used in replacement strings.

Matching the Beginning/End of a Line with "|<" and "|>"

The code "|<" occurring at the beginning of a search string ensures that the entire search string only matches text occurring at the beginning of a line.

Similarly, the code "|>" occurring at the end of a search string ensures that the entire search string only matches text occurring at the end of a line.

Unlike "|**L**" and "|**N**", "|<" and "|>" do not include the newline character(s) in the matched text. This is an important distinction when performing a replacement - with "|**L**" and "|**N**" the newline character(s) will be replaced; with "|<" and "|>" they are not replaced. With **{CONFIG, File Handling, File type}** set to record mode, these codes match the beginning/end of a record.

Matching Multiple Characters with "|M" and "|Y"

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. Select **{SEARCH, Search}** and enter the following search string:

"|M" (That's four characters)

The cursor should be at the beginning of the first quotation. Each time you press **{SEARCH, Next}**, the cursor should move to the next quotation. If it does not, 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:

```
Search:  Besides|Mparagraph:
Replace:                        (None, just press <Enter>)
```

The "|**M**" code often matches too much text. 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.

```
Search:  |W;|Y|>
Replace:                (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|LMOV BH,DH
```

Notice that substituting "|**M**" for "|**Y**" would not be the same — 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 following search strings will find a vowel or a consonant:

|{a,e,i,o,u} (vowels)
|{b,c,d,f,g,h,j,k,l,m,n,p,q,r,s,t,v,w,x,y,z} (consonants)
|!|{a,e,i,o,u,_,|d,|s} (consonants)

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 following search string:

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

matches "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".

Pattern sets may be embedded within each other for even more sophisticated searching. For example, the revised search string:

the |{tall|[ish],short|[ish],fat,thin} man

also matches "the tallish man", "the shortish man".

The search string:

|{|{foot,basket,base}ball,soccer,golf,hockey}

matches "football", "basketball", "baseball", "soccer", "golf" and "hockey".

The following search strings find numbers between 1 and 4 digits in length. They illustrate that there often are several equivalent ways to use pattern sets:

|{|d|d|d|d,|d|d|d,|d|d,|d}
|{|d|[|d]|[|d]|[|d]}
|{|d|[|d|[|d|[|d]]]}

(Note: Items that are substrings of another item must be placed after the larger item inside a pattern set.)

Using Text Registers in Search Strings

The contents of a text register can be used as part of a search string. The register contents are accessed with the pattern matching code "|**@(*r*)**" where '*r*' is the name of the register. 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
|@(1) night for a walk
|@(1) night for |@(2)

Regular Expressions

Regular expressions are a type of text pattern matching originally developed in the UNIX environment. They can be used as an alternative to VEDIT's normal pattern matching. The UNIX specifications are followed very closely. To use regular expressions they must be enabled either by setting **{CONFIG, Search options, Default search mode}** to "2" or "3", or by selecting "**( ) Reg-Exp**" in the Search or Replace dialog box.

The choice between normal 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, regular expressions are 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***.
- If you already know regular expressions from the UNIX environment, you don't have to learn another searching language.

Disadvantages of Regular Expressions:

- Regular expressions can be lengthy and verbose. For example, the pattern matching code "|A" is equivalent to the regular expression "[a-zA-Z]".
- Regular expressions are less flexible for searching multi-line patterns.
- Many characters have a special meaning in regular expressions. A special syntax must be used when searching for these characters.
- For the most part, regular expression searching is slower than normal pattern matching searching.
- Pattern matching has some capabilities not available in regular expressions. For example, "Pattern sets" can match optional items and any one of a set of items.

Regular Expression Basics

NOTE: To use regular expressions, be sure that the option "**( ) Reg-Exp**" is selected in the Search/Replace dialog box.

Many characters have special meaning in regular expressions:

^ (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).
. (period)	Simple wildcard that matches any character.

*****	Matches zero or more occurrences of the ***preceding*** character (or list).
+	Matches one or more occurrences of the ***preceding*** character (or list).
?	Matches zero or one occurrences of the ***preceding*** character (or list).

Examples:

o.e	Matches any text containing "o" followed by ***any character*** 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 ***zero or more*** 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 ***one or more*** 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 ***zero or one*** 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".

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 within square brackets "**[**" and "**]**" matches any ***one*** character in that list. A range of characters can be abbreviated using a hyphen "**-**". However, when the first character in the list is a "^" (caret) or "~" (tilda), the list matches any character ***except*** those in the list.

[abc]d	Matches any text containing "a", "b" or "c" followed by "d".
[a-z]	Matches all lower case letters.
[z-a]	A range may also be specified in reverse order.
[A-Za-z]	Matches all letters, upper or lower case.
[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 ***except*** for "a".
[^a-z]	Matches any character ***except*** for a lower case letter.
[~a-z]	Same. "**[~**" is equivalent to "**[**^".
[^0-9]	Matches any character except for a digit.
\[	Matches the "**[**" character.
****	Matches the "\" character.

The "**[ ] Case**" search option 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 select the **"[ ] Case**" option — you could use the expression "**[h][i]**" to search for the lower case word "hi".

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.

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.

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 in front of all non-alphanumeric characters.

In the replacement string, only the two characters "\" and "**&**" are special symbols.

Special Matching Characters

The following special matching characters are defined. They can be used in both the search string and replacement string of a regular expression.

\b Matches the ASCII backspace character (hex 08).

\d*DDD* Matches the character with decimal value '*DDD*'. All three digits ***MUST*** be present. "\d010" does not work in the search string; use "**\N**" instead.

\e Matches the ASCII <Esc> character (hex 1B).

\f Matches the ASCII Form-feed character (hex 0C).

\h*HH* Matches the character with hexadecimal value '*HH*'. Both digits ***MUST*** be present. "\h0A" does not work in the search string; use "**\N**" instead.

\n Matches the Line-Feed character (hex 0A). This is the "newline" character for UNIX type text files. To search for multiple-line patterns, use "**\N**" instead.

\N Matches the "newline" character(s) and allows searching for multiple line patterns. The "newline" depends upon the current file type and can be <CR><LF>, <LF> or <CR>. (Currently, "\N+" and "\N*" are not supported.)

\o*OOO* Matches the character with octal value '*OOO*'. All three digits ***MUST*** be present. "\o012" does not work in the search string; use "**\N**" instead.

\r Matches the ASCII Carriage-Return character (hex 0D).

\s Matches the ASCII space character (hex 20).

\t Matches the ASCII Tab character (hex 09).

\0 Matches the ASCII Null character (hex 00).

\@(*r*) Use the contents of text register '*r*' in this position in the search (or replace) string.

The "OR" Operator

The special character "|" (the "pipe" character, which is <Shift>-\ on the keyboard) 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 — "**\1**" through "**\9**". The groups are numbered in the order of their opening "**{**". Groups may also be nested. Groups may be referenced in either a latter part of the regular expression or in the replacement string. This allows portions of the matched text to be used as parts of the replacement text.

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".

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

The order of precedence of the operators is:

\	Highest
[ ]	
* + ?	
{ }	
Concatenation	
|	Lowest

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. Be sure that the option "**( ) Reg-Exp**" is selected in the Search/Replace dialog box.

```
Search: [A-Z][a-z]*
```

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

```
Search: 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.

```
Search: ^{.+}$\N\1$
```

Search for a line that is duplicated on the next line.

```
Search: .*,
Replace:                (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.)

```
Search:  {[Hh]}ello
Replace: \1i
```

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

```
Search:  [Hh]ello
Replace: &~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 "**&**".

Matching the "Newline"

Regular expressions CANNOT match "newline" sequences at the end of a line. Therefore, the expression "**[^a]**" matches any character expect for "a" ***and*** a "newline" (single Line-Feed or Carriage-Return ***and*** Line-Feed). Similarly, the "*****", "**+**" and "**?**" operators stop matching when they reach a newline.

[\h00-\h1f] Matches any control characters except "newline".

a.* Matches the letter "a" and all following characters up to, but not including, the newline.

The only exception to this is "**\N**" which explicitly matches the newline character(s).

end\Nbegin Match "end" at the end of a line followed by "begin" at the beginning of the next line.

Maximize Regular Expression Matching

Regular expressions originally came from the UNIX environment and we have made every attempt to follow the UNIX definitions as closely as possible.

Another ***important rule*** about UNIX 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".

Although this is a useful and powerful characteristic of regular expressions, it is not always desirable. (It is also not intuitive.) VEDIT lets you select either minimized or maximized regular expression matching in the Search dialog box. The default can be set with **{CONFIG, Search options, Default search mode}**.

When minimized, it would have matched "22" instead of "12231223" in the previous example. As another example, consider the search string:

a.+b

and the text:

12a3456b7890b

When minimized, it will match "a3456b"; when maximized, it will match "a3456b7890b".

Using maximized regular expressions 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 characteristic that the original UNIX designers decided to restrict searches to a single text line — a multi-line search for "{.*}\1" in even a small 1K file can involve over a million comparisons.

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.

Word Wrap and Right Margin When "word wrap" is enabled, words that 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>**. By default, the right margin is the width of the current window, but any value can be set.

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 **{EDIT, Center line}**.

Unlike dedicated word processors, VEDIT's word processing functions never insert special "control codes". Paragraphs are formatted by inserting only "newline" characters (Carriage-Return and/or Line-Feed depending upon the file type). 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.

Definition of "Word" and "Paragraph"

VEDIT considers a "word" to be any sequence of characters separated from each other by certain characters:

- A space always separates words from each other.
- Any control character including Carriage-Return, Line-Feed and Tab separates words from each other.
- A space always separates words from each other.
- Any control character including Carriage-Return, Line-Feed and Tab separates words from each other.

- The characters configured with **Config_String(WORD_SEP)** separate words from each other. By default, only the following characters are configured as separators:

  `, ; : ( )`

- The characters "(< [{" are treated as the beginning of a word. Similarly, ") >] }" are treated as the end of a word.

With the default **Config_String(WORD_SEP)**, the "< [{" and ">] }" characters are treated as part of the word. However since "(" and ")" are word separators, they are not treated as part of the word. This makes it easy to delete the words inside parentheses without deleting the parentheses themselves.

- As a special case, numbers with embedded commas, such as "10,000" are always treated as one word.

- The characters configured with **Config_String(WORD_SEP)** separate words from each other. By default, only the following characters are configured as separators:

By default, words are allowed to have embedded periods in them, as in "i.e.". Chapter 8 describes how to change the configuration parameter **Config_String(WORD_SEP)** in the **`vedit.cfg`** file.

To VEDIT, a paragraph is one or more lines of text separated from other paragraphs by these rules:

- A blank line always separates paragraphs from each other. The blank line is allowed to have "invisible" spaces and Tab characters.

- Lines beginning with the characters configured with **Config_String(PARA_SEP)** separate paragraphs from each other. By default, the following characters are configured as separators:

  `. @ ! \`

Lines starting with these configurable characters are assumed to be "print formatting commands" for programs like WordStar, Ventura Publisher and our V-PRINT. Print formatter command lines are not considered part of any paragraph and are therefore never re-formatted.

Note that just an indented line is not enough to separate one paragraph from another.

Indenting Text (Left Margin)

There are several ways to indent text so that it does not 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 automatically indent each new line of text for you. This is useful in word processing for indenting entire paragraphs, and for editing programs written in structured languages such as C, Pascal and Java.

The left margin determines how much newly entered lines will be indented. Normally the left margin is set to column 1.

You can change the left margin with **{CONFIG, Word processing, Left margin}**. However, it is usually easier to change the left margin on-the-fly by pressing **{EDIT, Indent}** (**<F8>** or toolbar), which increases the left margin, and **{EDIT, Undent}** (**<F7>** or toolbar), which reduces the left margin. These functions change the left margin by the *indent increment*, typically 4. The increment is set with **{CONFIG, Programming, Indent increment}**.

VEDIT indents text by padding lines with Tab characters and spaces. The optimum number of tabs and spaces will be used depending upon the currently set tab stops. If you prefer text to be indented using only spaces, set **{CONFIG, Tab/Fill, Expand <Tab> with spaces}** to "1", "3", "5" or "7".

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 **{EDIT, Indent}** and **{EDIT, Undent}**. 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. **{CONFIG, Printer, Left margin}** lets you set a separate printer left margin for this purpose.

Word Wrap (Right Margin)

"*Word wrap*" is a feature that 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.

To enable word wrap and **{EDIT, Format paragraph}**, you must enable **{CONFIG, Word processing, Enable word wrap}**. The desired right margin is set with **{CONFIG, Word processing, Right margin}**. The default value of "0" sets the right margin to the current window width, but any desired value can be set, even values wider than the window.

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

The right margin can be greater than the window width, in which case VEDIT will either scroll the window 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 with **{EDIT, Format paragraph}** (**<Ctrl-B>** or toolbar).

IMPORTANT NOTES: Word wrap should be disabled 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.

Do not 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

{EDIT, Format paragraph} formats (or re-formats) a paragraph to fit it between the current left and right margins.

A paragraph is formatted by placing as many words on each line as possible without exceeding the right margin. 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 a paragraph, first ensure that the left and right margins are set as desired and that **{CONFIG, Word processing, Enable word wrap}** is enabled. Then place the cursor anywhere in the paragraph and select **{EDIT, Format paragraph}** (**<Ctrl-B>** or toolbar). After formatting, the cursor moves to the next paragraph. Therefore, you can repeatedly select **{EDIT, Format paragraph}** to format one paragraph after another.

{CONFIG, Word Processing, Format paragraph options} controls how paragraph formatting deals with extra space characters at the ends of lines

NOTES: 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!

The supplied file KEY-MAC.LIB contains a key-stroke macro for formatting a columnar block of text. This lets you format just the words in a range of columns instead of all words on the line.

Offset Paragraphs

An "offset paragraph" is a 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.

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 justified 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 jagged. To unjustify a paragraph, disable **{CONFIG, Word processing, Justify paragraphs}**, set **{CONFIG, Word processing, Format paragraph options}** to "0" (or "4") and reformat the paragraph.

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

Format Paragraph Options

The default value of "0" for **{CONFIG, Word processing, Format paragraph options}** removes all extra spaces from a paragraph when it is formatted. It trims trailing spaces, removes extra spaces from between words and leaves only a single space following ".", "!" and "?". If this is not desirable for your application, you can select other options.

{CONFIG, Word processing, Format paragraph options} combines three options into one by having you add "mask" values (setting bits) for each desired sub-option.

Mask 1 Add a trailing space after each paragraph line except the last. This is needed for applications that ignore single "newlines" and would otherwise concatenate the words from two lines together. This extra space is allowed to exceed the right margin.

Mask 2 Allow any number of extra spaces between words.

Mask 4 Allow two spaces after ".", "!" and "?". Use this if you like two spaces between sentences.

For example, if you want trailing spaces after each line and two spaces between sentences, configure this parameter to "5".

Editing Multiple Files

You can simultaneously edit up to 32 files at a time in VEDIT. It is exceptionally flexible in how the files are displayed. Although each file typically has a corresponding window, a file can also be displayed in two or more windows. This is useful for editing two regions of the file at the same time, or for editing the file in two modes at the same time, e.g. in ASCII and Hexadecimal.

In Windows, you can start up VEDIT without any initial files, or you can "drag and drop" one or more files to the VEDIT icon to start up VEDIT with the files already loaded. Of course, you can also open and close files from inside the editor.

It is easiest to edit additional files using these "hot-keys" in the "normal" keyboard layout.

<Ctrl-O> Opens a new file for (simultaneous) editing in a new buffer. As a convenience, if the specified file does not exist, it is created. It also opens a corresponding window. This is the most common way of editing additional files and creating new files. Hot-key for **{FILE, Open}**.

<Alt-O> Opens a new file for editing in the current (same) buffer. You are 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. Hot-key for **{FILE, Open (More), Same buffer}**.

<Alt-Y> Similar to **<Ctrl-O>** except that it opens the new window by splitting the current window horizontally into two windows. Hot-key for **{FILE, Open (More), Horizontal window}**.

<F5>
<F6> These keys toggle, round-robin fashion, to the previous/next open buffer (file). This is the most common way of switching between files. Hot-keys for **{FILE, Previous buffer}** and **{FILE, Next buffer}**; they are also available on the toolbar.

<Ctrl-F4> Closes the current window. If it is the last window attached to the file (buffer), it also closes the file, prompting whether the file is to be saved or abandoned. However, it does not close the last window and buffer; it then only closes the file. Hot-key for **{WINDOW, Close}**. It is the same as clicking the window's "close" button.

HINT: You can open two or more files at once by entering their names in the "Filename:" prompt in the File-Open dialog box. You can also specify a "Save as" name or begin editing on any desired line number. See the on-line help for this dialog box or **{File, Open}** in Chapter 6 (Menu Reference) for more information.

{FILE, Open (More)} Sub-menu

VEDIT has three variations of **{FILE, Open}** to save you steps when editing multiple files.

Many times you will edit one file after another without needing to switch back and forth between them. You can do this in three ways:

- You can simply open each additional file. However, after a while you will have so many files and windows open that the screen gets cluttered and things get confusing.
- As you finish with each file, you can close it and then open the next file. This keeps things simpler.
- Use **{FILE, Open (more), Same buffer}** (**<Alt-O>**). It is a shortcut way of closing the current file and opening the next file. It saves keystrokes and mouse clicks.

Sometimes you want to see two files side-by-side or one above the other, perhaps to compare them. You can do this in two ways:

- Open both files with **{FILE, Open}**. Then resize the windows so that they are side-by-side or above each other.
- Open the first file with **{FILE, Open (more), Same buffer}**. By default, VEDIT displays it in a full-sized window. Open the second file with **{FILE, Open (more), Vertical window}** or **{FILE, Open (more), Horizontal window}**. The screen will be nicely split into two windows, saving you the time of manually resizing windows.

Once a file has been opened in a window, the window can be moved or resized in the usual manner with the mouse or by selecting "Move" or "Size" from the window's option menu (click on the window's icon or press **<Alt-Space>** and **<Cursor Right>**).

Select **{WINDOW, Cascade}** to display all windows at once with overlapping, or **{WINDOW, Tile}** to display all windows without overlapping.

Selecting **{VIEW, Zoom}** is the same as clicking on the Window's zoom button. It zooms the current window to it maximum size for easier editing. Other windows you switch to will also be zoomed. Select it again to restore the windows to their normal size.

Alternatively, select **{VIEW, Full size}** to expand the current window to full-size without zooming all windows. Or select **{VIEW, Full size - All}** to expand all windows to full-size. This is convenient if you want to view most, but not all, windows at full-size.

{VIEW, Reset} resets all windows by restoring their normal color and display mode. Each open buffer (file) is displayed in just one window; additional windows are removed. The windows are displayed as cascaded or full-size, depending upon the setting of **{CONFIG, Display options, Auto-create window style}**.

SUGGESTION: To clean up a cluttered window display, try **{WINDOW, Cascade}** or **{VIEW, Full size - All}**. If the windows are also in unwanted colors and/or display modes, first select **{VIEW, Reset}.**

Optional Configuration

{CONFIG, Display options, Auto-create window style} determines if newly created window are initially "full-size" and completely overlapping, or are smaller and cascaded. The default is "cascaded" because this is typical for other windows programs.

However, most users prefer full-size windows; this lets you edit each file in a large window; it has some advantages to zooming the window.

See the heading "Full-size Windows" below.

Switching Between Files (Edit Buffers)

You can switch between the files currently being edited by toggling between them, by switching directly to a particular buffer, or by using "point and shoot" selection.

➢ **To switch to another file:**

1. Select **{FILE, Previous buffer}** or **{FILE, Next buffer}** (**<F5>**, **<F6>**). This toggles, round-robin fashion, to the next/previous open edit buffer and the file (if any) being edited in it.

   These functions are also available on the toolbar.

-OR-

1. Select **{FILE, Buffer switch}** (**<F4>**). You are prompted for the buffer number:

   ```
   Buffer number:
   ```

2. Enter the desired buffer number, or select the desired buffer using point & shoot.

Closing Files and Windows

The easiest way to close a file, without exiting VEDIT, is to double click the window's icon; this is the same as selecting **{WINDOW, Close}** (**<Ctrl-F4>**). If this is the only window displaying the file, which is the typical case, it closes the file and the corresponding buffer.

If a file is displayed in two or more windows, **{WINDOW, Close}** does not close the file until you close its last window.

Alternatively, you can select **{FILE, Close}** to close the current buffer, saving or abandoning any file in it. It also closes all windows attached to the buffer. In the typical case where each file is displayed in only one window, **{FILE, Close}** and **{WINDOW, Close}** are identical.

NOTE: VEDIT always keeps one buffer and window open. If you attempt to close the last window, it only closes the file. The last buffer and corresponding window will remain open, but will be empty.

Use **{FILE, Exit}** when you are ready to exit VEDIT. It will let you selectively save or abandon each modified file, or save/abandon all files.

➢ **To close the current file (buffer):**

1. Select **{WINDOW, Close}** (**<Ctrl-F4>**) or double-click the window's icon. Alternatively, select **{FILE, Close}**.

   If the current file has been modified and not yet saved to disk you are prompted with:

   ```
   Save current file? [Yes]  [No-Abandon]
   ```

2. If you select [Yes], VEDIT will save the current file with all modifications. If you select [No], VEDIT will abandon changes that have been made since the last time the file was saved. In either case, it also closes the current buffer and buffer. (However, VEDIT always keeps one buffer and window open; they may be empty.)

3. If you select [Yes] to save the text, but no file is open, you will be prompted for the name under which to save the file:

   ```
   No filename specified! Enter "Save As" filename:
   ```

   Enter the desired name for the file.

MOUSE: You can also close a file by double-clicking the mouse on the window's left side icon. This is equivalent to **{WINDOW, Close}**.

{WINDOW, Remove} lets you close any window without first switching to it. Its primary purpose is to close (remove) special "command mode" windows that were created in the VEDIT PLUS macro language or with **{WINDOW, Split}**.

Copying Text From One File to Another

A common reason for editing two files is to copy portions from one file to another. This requires the use of a text register.

➢ **To copy text from one file to another:**

1. Highlight the block in the normal fashion. Either a stream, line or columnar block can be copied. You may want to use **{BLOCK, Set stream marker}** (**<F9>** or the toolbar) to highlight a very large block.

2. Select **{BLOCK, Copy to register}** (default: **<Numpad+>**). Enter the desired text register number; we suggest "0" through "9". You can immediately press **<Enter>** or **<Numpad+>** again to select register "0", also called the *scratchpad*.

3. Switch to the second file and move the cursor to the desired position.

4. Select **{BLOCK, Insert register}** (default: **<Numpad*>**). Select the same text register as in step 2.

If the block is too large to fit in a text register, you will receive an error message. See "Cut & Paste Huge Blocks" earlier in this chapter for a work-around; one way is to use the Windows clipboard.

Starting (Default) Directory for File-Open

VEDIT gives you flexibility determining in what directory most file selection dialog boxes start. Starting in the correct directory can save a lot of time. For some dialog boxes, VEDIT selects the most likely directory. Others start in the directory of the current file, or optionally in the "*current*" directory, which can be changed.

By default, the starting directory for **{FILE, Open}**, **{FILE, Save as}**, **{EDIT, Insert file}** and **{BLOCK, Write to disk}** is the same directory as the file in the current buffer. If the current buffer has no file open, e.g. following **{FILE, New}**, the starting directory is the "current" directory.

These file selection dialog boxes include a **[ ] Change directory** option. Initially, the box is not enabled and the dialog box starts in the same directory as the file in the current buffer.

If you check the **[ ] Change directory** box, the last directory you select in the dialog box becomes the new "current" directory. Also, these dialog boxes will then start in the "current" directory.

HINT: When editing multiple files in different directories, first switch to a file which is in the same directory as the next file you want to open. **{FILE, Open}** will then start in the desired directory.

{MISC, Load/Execute macro} starts in the *VEDIT Home Directory*, e.g. `c:\vedit`, because macro files are usually located there. Similarly, **{CONFIG, Save / Load config}** starts in the *User Config Directory* because the user's configuration files are located there.

The initial "current" directory for the Windows version of VEDIT is determined as follows:

1. If VEDIT is started by dragging and dropping a file on its icon, the directory containing the file becomes the current directory.
2. If VEDIT is started by clicking its icon, the current directory can be selected by setting the icon properties' "Start in" field.

   However, the first file opened with **{FILE, Open}** or by dragging and dropping a file onto a VEDIT window will set the current directory.
3. If the Windows or DOS version of VEDIT is started from the DOS prompt, the "current" directory is the current DOS directory.

Edit Buffer Details

VEDIT has 32 available edit buffers, each of which can have one file open for editing. The edit buffers are always in one of three possible states:

- Closed. We also say that a closed edit buffer is "*available*" or "*unused*".
- Open without a file. The edit buffer may or may not contain text. If the buffer contains no text, we say that it is "*empty*".
- Open with a file open. The edit buffer is being used to edit a file.

Edit buffers are normally used to edit files, but can also be used as editable "scratchpads" that have no file open. You can open an empty edit buffer without a file by selecting **{File, Buffer switch}** (**<F4>**) and entering the number of an unused buffer. Or you can select **{FILE, New}** to open the next available buffer. You can then perform any normal editing operations in this buffer.

Once an edit buffer is opened, it remains open until you explicitly close it with **{FILE, Close}** or **{WINDOW, Close}**. However, VEDIT always keeps one buffer and window open. If you attempt to close the last buffer, it only closes the file. The last buffer and corresponding window will remain open, but will be empty.

The contents of an edit buffer can be inserted into another buffer (file) just like a text register. However, a block of text cannot be copied to another buffer — ***you cannot change the contents of an edit buffer except when it is the "active" buffer***. This limitation prevents you from accidentally altering any file open in the buffer.

➢ **To insert another edit buffer into the current buffer (file):**

1. Position the cursor at the desired location in your file. The buffer contents will be inserted just before the cursor.
2. Select **{BLOCK, Insert register}** (default: **<Numpad*>**). You will be prompted with:

   ```
   Register number:
   ```

3. Enter the buffer number followed by "**+buffer**". For example, to select buffer 5, enter "**5+buffer**".

NOTES: You should only insert an edit buffer that either has no file open or is reasonably small. In particular, if the open file is large, only the portion currently in memory will be inserted.

Use **{EDIT, Insert file}** to insert another file into the current file.

Windows

While windows and buffers may appear to be equivalent during routine editing, VEDIT's windows and buffers are actually completely independent of each other. It is possible to have buffers without windows and windows without buffers; this is especially true in the VEDIT PLUS macro language.

This advanced topic describes how buffers and windows interact.

Introduction

VEDIT can have up to 32 edit buffers open at once; they are numbered "1" through "32". Each buffer can have a file open in it for editing, although it is not necessary for a buffer to have an open file. In other words, multiple "<<No name>>" buffers can be open at once. At least one buffer is always open; although the file in the last buffer can be closed, the last buffer itself cannot be closed.

When you open additional files with **{FILE, Open}**, additional buffers are automatically opened, as needed, to hold the files.

VEDIT can also have up to 32 windows open at once; they have an ID number between "1" and "32". These are sometimes referred to as "editing" or "Visual mode" windows to distinguish them from special "Command mode" windows. Command mode windows have a non-numeric name such as "$" or "H". They are often created and used by the VEDIT PLUS macro language.

Windows are attached to buffers

To display buffers in windows, VEDIT automatically "*attaches*" a window to a buffer. When you display one buffer (file) in two or more windows, multiple windows are attached to one buffer.

Each time a new buffer is opened in Visual Mode (e.g. when you open a file), a new window is also created and attached to the buffer. The window is initially full screen in size and overlaps any other windows.

You might assume that buffer 1 is always displayed in window 1, buffer 2 in window 2 and so on. This is not true! For example, if you initially have one file open and then select **{WINDOW, Split}**, buffer 1 will be attached to windows 1 and 2. When you then open a second file, buffer 2 will be attached to the auto-created window 3.

Fortunately, it is rarely important to know the window's ID number.

How window names are displayed

There are two ways of identifying windows:

- Each editing window has a unique ID number such as "1" or "2". Special "Command mode" windows have a non-numeric name such as "$" or "H". This ID/name can be displayed in angle-brackets, e.g. **<2>** or **<$>**.

- Each editing window is attached to an edit buffer. The displayed name is the associated buffer number, e.g. **[2]**. Multiple windows per buffer are displayed as e.g. **[2:1]** and **[2:2]**.

By default, VEDIT displays both the ID number and the buffer number; however, if they are the same, it only displays the ID number. Therefore:

<3> [2] Indicates this is window "3" which is attached to buffer "2".

[1] Indicates this is window "1" which is attached to buffer "1". It does not display "<1>", because it is redundant.

<$> Indicates this is the special "Command Mode" window. It is never attached to a buffer.

To handle special applications and personal preferences, **{CONFIG, Display options, Window name display style}** can be changed to always display the window ID, the buffer number, or both.

The point and shoot for **{WINDOW, Switch}** displays each window's ID number, the buffer number and the filename.

Switching Between Windows

When each buffer (file) is displayed in its own window, select either **{FILE, Next buffer}** (**<F6>** or toolbar) or **{FILE, Previous buffer}** (**<F5>** or toolbar) to toggle round-robin between the files being edited.

Or you can select **{FILE, Buffer switch}**B (**<F4>**) or **{WINDOW, Switch}** (**<Alt-F5>**) to switch directly to a desired buffer or window.

When a buffer is displayed in two or more windows, you must use **{WINDOW, Next window}** (**<Ctrl-F6>**), **{WINDOW, Previous window}** (**<Ctrl-F5>**) or **{WINDOW, Switch}** (**<Alt-F5>**) to switch to the additional windows.

You can also switch to any visible window by clicking the mouse anywhere within the window; however, the desired window will often be covered by other windows.

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:**

1. Select **{WINDOW, Zoom}**.

The zooming remains in effect until you select **{WINDOW, Zoom}** again to "de-zoom" the screen to redisplay all windows.

MOUSE: You can also zoom a window by clicking the mouse on the "zoom" button. The button then changes to a "de-zoom" button.

"Full-Size" Windows (Windows version)

VEDIT has the useful concept of "full-size" windows. This is similar to you manually stretching a window's borders to the maximum possible size. Moreover, a full-size window will remain full size if you change VEDIT's overall window size, toggle the toolbar on/off or toggle the command mode window on/off.

A window remains full-size until you explicitly resize, tile or cascade it.

A full-size window is not the same as a "zoomed" window, even though they look similar and serve similar purposes. When you zoom a window, each window you switch to will also be zoomed, until you "dezoom". In contrast, some of your editing windows can be full-size, while others are custom sized or tiled.

{CONFIG, Display options, Auto-create window style} determines if newly created window are initially full-size or are smaller and cascaded. The default is "cascaded" because this is typical for other windows programs.

SUGGESTION: We suggest setting **{CONFIG, Display options, Auto-create window style}** to "1" to create new windows as full-size windows. This lets you edit in the largest possible window without having to zoom all windows.

Once you are familiar with VEDIT, we are confident that you will will use and appreciate the "full-size" window concept.

Editing One File in Two Windows

VEDIT can display a file in more than one window at a time. You can thereby view different sections of a file at the same time. For example, you could refer to definitions listed at the beginning of a file while editing in the middle of the file.

There are several ways to get a file displayed in two windows at once:

- **{WINDOW, Split}** splits the current window into two windows. The current file will initially be displayed in both the current and new window.
- **{VIEW, Toggle hex mode split}** splits the current window vertically; the left windows displays in hexadecimal; the right window displays in ASCII. Unlike the other splits, the cursors in both windows move together.

All windows displaying a common file update together 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.

Chapter 5

Advanced Topics

This chapter covers these topics:

- Covers the **`startup.vdm`** file in detail.
- Describes how you can add your own custom editing functions to the **{USER}** and **{TOOL}** menus.
- Covers the File-type specific configuration, Color syntax highlighting, Template editing and HTML editing features in detail.
- Introduces "Command Macros".
- Gives step-by-step directions on how to use the supplied macros PRINT, WILDFILE, COMPARE, COMPDIR, SORT, DBASE and CFUNC.
- Explains how to set up and use the "Ctags" symbol lookup feature.
- Introduces VEDIT's compiler support. The on-line help describes it in complete detail.

STARTUP.VDM File

The **`startup.vdm`** file is a special command macro which VEDIT executes upon startup. It sets up many of VEDIT's advanced features, including the **{USER}** and **{TOOL}** menus, file-type specific configuration, template editing and color syntax highlighting.

The **`startup.vdm`** file can easily be modified to optionally set up special hot-keys and force any desired configuration settings. Having a **`startup.vdm`** file is highly recommended but optional, and no error is given if the file is not found.

Since **`startup.vdm`** is a command macro, it can be modified in any way desired, especially if you are familiar with the macro language. However, you don't need to fully understand the macro language to modify **`startup.vdm`** to your preferences.

NOTE: The best way to understand this topic is to open the supplied **`startup.vdm`** file with VEDIT and examine it.

If you do not have **`startup.vdm`** in your *User Config Directory* (typically "c:\vedit"), simply copy **`startup.org`** to **`startup.vdm`**. We supply **`startup.org`** so that you can easily restore our default startup file.

The **`startup.vdm`** file is organized into several sections. It has these capabilities:

- Set up color syntax highlighting and template editing. Also set up other file-type specific configuration parameters. These features can then be enabled with **{CONFIG, Programming, File-type specific config}**.
- Set up the **{USER}** menu from the file **`user.mnu`**. A later topic describes how to modify this file with your own custom editing functions.
- Set up the **{TOOL}** menu as the **{Tutorial}** menu so that you can easily access tutorial topics. Alternatively, place the Compiler Support functions into the **{Tools}** or **{JavaTools}** menu.
- Set up the hot-key to open the file **`mynotes.txt`** in a pop-up window. The default hot-key is <**Alt-0**>.
- Optionally enable the hot-key for "ctags" lookup support.
- Optionally add other keystroke macros to the keyboard layout. These will override any assignments in **`vedit.key`**.
- Optionally force any desired configuration settings. These will override any settings in **`vedit.cfg`**.

Most functions performed by **`startup.vdm`** are optional; you must edit the file to enable the function. Typically, you only need to delete some "//" to enable the function.

For example, **`startup.vdm`** contains the following line which, if the initial "//" are removed, assigns <**F12**> to execute the "ctags" lookup macro by selecting **{MISC, More macros, UTAGS}**.

```
// Key_Add(F12",'[MENU]MMU',OK)
```

It also contains the following line which, if the "//" are removed, enables VEDIT to auto-save all modified files every 15 minutes:

```
// Config(F_AUTO_SAVE,15)
```

The following headings describe the features that can be set up with **`startup.vdm`** in more detail.

UNIX: ***QNX:*** The UNIX and QNX versions start up by executing the file "veditrc" in the *VEDIT Home Directory*. This macro file contains the commands **Config_Load("vedit.cfg")** and **Key_Load("vedit.key")** to configure VEDIT to the preferences of each user. The documentation supplied with the QNX and UNIX versions describes this in more detail.

Changing Configuration with STARTUP.VDM

You can set any desired configuration parameters with **Config( )** commands in the **`startup.vdm`** file. Since the **`startup.vdm`** file is executed after **`vedit.cfg`** is loaded, ***they will override the configuration saved with*** **{CONFIG, Save config}**.

Although this can get confusing, it has some advantages. For example, you may have configured VEDIT to auto-save your changes every 15 minutes. However, one day, you temporarily turn off this feature. With **{CONFIG, Auto-save config}** normally enabled, this change is saved. You then forget to turn this feature back on; most likely you won't notice your mistake until you lose some work.

This is an example of why you might want to set important configuration settings in **`startup.vdm`**. Any configuration changes you make in VEDIT will then be temporary, even if you "save" them.

Obviously, this can confuse a novice VEDIT user, because it will appear that certain configuration settings cannot be saved. For this reason, our supplied **`startup.vdm`** file does not override any configuration settings.

However, the supplied **`startup.vdm`** contains this and other configuration commands that have been disabled by placing "//" in front of the command. ("//" forces the remaining line to just be a comment.)

You can easily enable any of these configuration commands or add your own. The on-line help topic "Configuration" lists all of the **Config( )** commands.

- **Configure "auto-file save" feature in `startup.vdm`:**

  1. Open the file **`startup.vdm`** in the *User Config Directory* for editing.
  2. Locate the "// Config(F_AUTO_SAVE,15)" line.
  3. Delete the leftmost "//" to activate the command.
  4. Save the file and exit VEDIT.
  5. Restart VEDIT and notice that **{CONFIG, File handling, Auto-save interval}** is set to 15 minutes.

  No matter how you change the "Auto-save interval", each time you start VEDIT, it will be set back to "15".

Using a different startup file

You can specify a startup file other than **`startup.vdm`** with the "**-i**" invocation option. You can start up VEDIT without **`startup.vdm`** or any other startup file with the invocation option "**-i** *xxx*", where '*xxx*' is a non-existent file. No error message will be given.

vpw -i *execfile*	'*execfile*' is executed as the startup file in place of **`startup.vdm`**.
vpw -i xxx	Invoke VEDIT without any startup file.

HINT: "**-i** *xxx*" is useful for debugging startup configuration problems.

Invocation options can be specified in Windows by changing the icon's properties, by using the "Run" command, or by invoking VEDIT from the DOS/NT command prompt. See "Starting VEDIT for Windows" in Chapter 4.

VEDIT looks for **`startup.vdm`** first in the "current" directory, then in the *User Config Directory*, and finally in the *VEDIT Home Directory*. Except for Network and special installations, the *User Config Directory* and the *VEDIT Home Directory* are the same, e.g. "c:\vedit".

Since VEDIT first looks in the "current" directory, you can set up custom configurations for different projects by having different **`startup.vdm`** files in each project's directory. However, this is only practical if you invoke VEDIT (Windows or DOS version) from the DOS/NT command prompt.

You can create different VEDIT icons for different projects or tasks by using the "**-i**" invocation option to specify the desired startup file. Or you might create different icons for different users that share one computer. Here are some examples that could be used as the "Target" in the VEDIT icon's properties: (It is called "Command Line" in Windows 3.1)

c:\vedit\vpw.exe -i startup2.vdm

c:\vedit\vpw.exe -i c:\project\startup.vdm

c:\vedit\vpw.exe -i toms.vdm

Name of STARTUP.VDM and VEDIT.INI

We always refer to the name of the default startup file as **`startup.vdm`**, but you can change the name by editing the **`vedit.ini`** Windows parameter file.

Furthermore, you can specify a Windows parameter file other than **`vedit.ini`** with the "**-k**" invocation option:

vpw -k*inifile* '*inifile*' is used as the Windows parameter file in place of **`vedit.ini.`** (There must be no space between "-k" and the filename.)

This lets you completely customize VEDIT for different tasks. For each task, create a VEDIT icon which uses the "**-k**" invocation option to specify a different Windows parameter file. Each Windows parameter file can then specify a different startup macro file. One advantage of using this scheme over the "**-i**" option described above, is that each task can have a different set of recently used files in the **{FILE}** menu and can have different display and printer fonts.

{USER} and {TOOL} Menus

Two sets of custom editing functions can be added to the main menu as the **{TOOL}** and **{USER}** menus. As with all menu functions, the custom functions can have hot-keys assigned to them.

The supplied **`startup.vdm`** loads the file **`user.mnu`** as the **{USER}** menu. It also loads the file **`tutor.mnu`** as the **{TOOL}** menu, and renames the menu to **{Tutorial}**.

Both the **{USER}** and **{TOOL}** menus can have any desired name appear on the menu bar. In particular the **{TOOL}** menu is often renamed to reflect its purpose. The commands **Config_String(USER_MENU,"&User")** and **Config_String(TOOL_MENU,"&Tools")** set the menu names, which can be up to 16 characters long. The "&" indicates which character is underlined in the Windows version.

The default **{USER}** menu, loaded from **`user.mnu`**, includes some of the macros listed in Appendix E (Application Notes.) You can delete and add items to **`user.mnu`** as desired to create a custom **{USER}** menu.

The default **{TOOL}** menu, loaded from **`tutor.mnu`** and displayed as the **{Tutorial}** menu, contains over a dozen hands-on tutorial topics.

Different **{USER}** and **{TOOL}** menus can also be loaded with **{MISC, Load {USER} menu}** and **{MISC, Load {TOOL} menu}**. Although the default filename extension is "***.mnu**", it can have any desired name.

The editing functions in the **{USER}** and **{TOOL}** menus are implemented as command macros. Although VEDIT PLUS is needed to fully realize its potential, VEDIT users can modify the existing functions or add new ones by copying macros from **`key-mac.lib`** into **`user.mnu`**.

Examine the **`user.mnu`** file and note that each editing function consists of three parts:

Highlight Number	Determines which letter in the "Item name" is highlighted. This is usually the first letter unless several Item Names begin with the same letter. Follow the number with "+128" to display a divider in the menu above this item.
Item Name	The function's name as it will appear in the menu.
Command Sequence	The macro language command(s) to be executed when this item is selected. The commands must be listed on one line, but can be as long as needed.

The set of all custom editing functions for the **{USER}** menu is then loaded into special text register "124" with the command **Reg_Load(124, "user.mnu")**. Similarly, the **{TOOL}** menu is loaded with the command **Reg_Load(123,"tutor.mnu")**. These commands are typically included in the **`startup.vdm`** file.

For example, the following addition lets you run our V-SPELL spelling corrector from inside VEDIT: (Enter the last two lines as one line!)

```
5
Run V-SPELL
Reg_Set(103,PATHNAME) File_Close()
  Sys("vs |@(103)",DOS) File_Open(@103)
```

Notes:

If the contents of text registers 123 and 124 are not set up properly, the error "INVALID MENU" is displayed when you attempt to access the main menu. This also empties register 123 or 124 so that the main menu continues to work.

The command **Reg_Empty(124)** empties the **{USER}** menu. Similarly, **Reg_Empty(123)** empties the **{TOOL}** menu.

DOS: When the **{USER}** or **{TOOL}** menus are empty, they do not appear on the main menu bar.

The on-line help topic "USER" describes how to add "Toggle into VGA 132 Column Mode" and "Toggle to 80 Column Mode" functions to the **{USER}** menu.

File-type Configuration

The supplied **startup.vdm** file sets up "File-type specific configuration" which configures VEDIT according to common filename extensions. For example, when you open a ".c" file, auto-indenting is enabled, while with a ".txt" file, word wrap is enabled. It can also load a "Template Editing" macro for C or HTML editing and enable color syntax highlighting.

NOTE: The original **startup.vdm** file is also supplied as **startup.org**. If you have no **startup.vdm** file, simply copy **startup.org** to **startup.vdm.**

By default, this feature is disabled in order not to confuse new users.

➢ **To enable File-type specific configuration:**

1. Set **{CONFIG, Programming, File-type specific config}** to "1", "3" or "7". A value of "3" also enables template editing. A value of "7" also enables color syntax highlighting.
2. If desired, select **{CONFIG, Save config}** to ensure that the configuration change is permanent for the next time you run VEDIT. (This step in not needed if **{CONFIG, Auto-save config}** is enabled.)
3. Test the feature by opening a file with a ".c" extension. Check that **{CONFIG, Word processing, Enable word wrap}** is off.

   Open a file with a ".txt" extension. Check that **{CONFIG, Word processing, Enable word wrap}** is on.

-OR-

1. Edit the file **startup.vdm** in the *User Config Directory*.

   Search for "Config(PG_F_AUTO_CFG,1+2+4)" and delete the preceding "//" to enable the command.

   Config(PG_F_AUTO_CFG,1+2+4) //Enable file-type config
2. Save the file and exit VEDIT.
3. Restart VEDIT to process the new **startup.vdm** file. Check that **{CONFIG, Programming, File-type specific config}** is set to "7".
4. Open a ".txt" file. Check that **{CONFIG, Word Processing, Enable word wrap}** is on.

   Open a ".c" file. Check that **{CONFIG, Word Processing, Enable word wrap}** is off.

Although many of the macro language commands in **startup.vdm** are outside the scope of this manual, the following description will help you modify **startup.vdm** for your needs. (The documentation supplied with VEDIT PLUS describes all macro language commands in detail.)

NOTE: This topic will be easier to understand if you are viewing the **startup.vdm** file at the same time.

Several blocks of commands configure each edit buffer according to the filename extension. For example, here are the commands that check for a ".c" or ".h" file:

```
// Test if this is a ".C" or ".H" file...
//
BOF()
if ( Search(".C|>",NOERR)==1||Search(".H|>",NOERR)==1){
 Buf_Switch(#109)
 Config(PG_AUTO_IND,1,LOCAL)
 Config(PG_IND_INC,4,LOCAL)
 Config(W_LF_MARG,0,LOCAL)
 Config(W_RT_MARG,0,LOCAL)
 Config_Tab(8;LOCAL)
 if (Config(PG_F_AUTO_CFG)&2) {
  Config(PG_TEMPLAT,1,LOCAL)   //Template editing for C
 }
 if (Config(PG_F_AUTO_CFG)&4) {
  Config(PG_E_SYNTAX,1,LOCAL)   //Syntax highlight for C
 }
 Goto ENDMACRO
}
```

The **Search( )** command identifies the desired filename extension. The commands between "**{**" and "**}**" are then executed.

Notice the **Config( )** commands which set the desired configuration parameters for this file type. You can add, remove and change these commands as needed. The on-line help topic "Configuration" describes all **Config( )** commands.

The **Config( )** commands must include the "LOCAL" option so that the configuration is only changed in the current buffer (file). Most of the configuration parameters found in **{CONFIG, Word Processing}** and **{CONFIG, Programming}** and the Tab stops are "edit buffer dependent". These parameters have a "(*) " in their name and can have a different value in each edit buffer. (See **{CONFIG, Config all buffers}** in Chapter 6 for more information.)

If you change a configuration parameter which is not *edit buffer dependent*, such as **Config(F_E_BACKUP,"Backup files")**, it will affect all files being edited. Any "LOCAL" option will be ignored.

The last two **Config( )** commands enable template editing and syntax highlighting if **{CONFIG, Programming, File-type specific config}** has these features selected.

You can add configuration for additional filename extensions by simply copying an existing block of commands and modifying it as needed. Change the **Search( )** command to the new filename extension. Then modify the **Config( )** commands as needed.

See also:

The topic "Startup.vdm File".
The topic "Template Editing".
The topic "Color Syntax Highlighting".
The topic "HTML Editing Features".

Color Syntax Highlighting

Primarily intended for program and HTML editing, syntax highlighting displays different logical parts of a program in different colors. For example, reserved words, comments, string and numeric arguments and special symbols can each be in colors different from the rest of the text.

The supplied syntax definition files include:

C.SYN	For C and C++
CLIPPER.SYN	For Clipper
COBOL.SYN	For COBOL
FOLIO4.SYN	For Folio View Flat File
HTML.SYN	For HTML (Web pages)
MASM.SYN	For Microsoft MASM Assembler
MBASIC.SYN	For Microsoft BASIC
PASCAL.SYN	For Pascal and Object Pascal
PERL.SYN	For Perl (preliminary)
REXX.SYN	For Rexx
SQL-PL.SYN	For SQL and PL/SQL
SYSTAT.SYN	For Systat product

➢ **To load a syntax highlighting definition file:**

1. Select **{MISC, Load syntax file}**.
2. Enter the desired filename or use point and shoot to select the desired file. The file selection dialog box defaults to all " ***.syn**" files in the *VEDIT Home Directory*.
3. If desired, change the colors in **{CONFIG, Syntax colors}**. Each of VEDIT's color schemes has some reasonable colors already set up.

Only one syntax definition file can be loaded into VEDIT at a time. VEDIT does not (currently) support syntax highlighting for C in one window and syntax highlighting for HTML in another window. (It's coming.)

You may want to disable **{CONFIG, Config all buffers}**. Syntax highlighting can then be enabled or disabled independently for each file with **{CONFIG, Programming, Enable color syntax highlighting}**.

Automatic syntax highlighting for C

The supplied **`startup.vdm`** file contains the commands to perform syntax highlighting (and template editing) for C files. This is part of the "File-type specific configuration". When you open a .C, .CPP or .H file, syntax highlighting is automatically enabled; for other file-types it is disabled. This automatic syntax highlighting must be enabled.

➢ **To enable automatic syntax highlighting:**

1. Set **{CONFIG, Programming, File-type specific config}** to "5" or "7". A value of "7" also enables template editing.
2. If desired, select **{CONFIG, Save config}** to ensure that the configuration change is permanent for the next time you run VEDIT. (This step in not needed if **{CONFIG, Auto-save config}** is enabled.)
3. Test the feature by opening a file with a ".c" extension. Check that **{CONFIG, Programming, Enable syntax highlighting}** is on.

   Open a file with a ".txt" extension. Check that **{CONFIG, Programming, Enable color syntax highlighting}** is off.

-OR-

1. Edit the file **`startup.vdm`** in the *User Config Directory*.

   Search for "Config(PG_F_AUTO_CFG,1+2+4)" and delete the preceding "//" to enable the command.

   Config(PG_F_AUTO_CFG,1+2+4) //Enable file-type config
2. Save the file and exit VEDIT.
3. Restart VEDIT to process the new **`startup.vdm`** file. Check that **{CONFIG, Programming, File-type specific config}** is set to "7".
4. Open a ".c" file. **{CONFIG, Programming, Enable color syntax highlighting}** should be on.

   Open a ".txt" file. **{CONFIG, Programming, Enable color syntax highlighting}** should be off.

Automatic syntax highlighting for HTML

The **`startup.vdm`** file is easily changed to support syntax highlighting for HTML instead of C. Open **`startup.vdm`** for editing and follow these steps.

➢ **To enable syntax highlighting for HTML:**

1. Disable the `Syntax_Load("c.syn")` command by preceding it with "//". Enable the `Syntax_Load("html.syn")` command by removing the preceding "//".

   // Syntax_Load("c.syn") // Setup syntax highlight for C
   Syntax_Load("html.syn") // Setup syntax highlight for HTML
3. Find "Config(PG_E_SYNTAX,1,LOCAL)" in the file-type specific configuration section corresponding to .C files. Disable the command by preceding it with "//".

   // Config(PG_TEMPLAT,1,LOCAL) //Template editing for C
4. Find "Config(PG_E_SYNTAX,1,LOCAL)" in the section corresponding to .HTML files. Enable the command by removing the preceding "//".

   Config(PG_TEMPLAT,1,LOCAL) //Template editing for HTML
5. Follow the steps above to enable automatic syntax highlighting.

See Also:

See the topic "HTML Editing" for detailed steps on editing the **`startup.vdm`** file to enable syntax highlighting.

Automatic syntax highlighting for other languages

You can add automatic syntax highlighting for other programming languages or file types to the **`startup.vdm`** file. Follow these general steps:

1. As described in the earlier topic "File-type specific configuration", add another block of commands for a new filename extension.
2. In the new block of commands, enable syntax highlighting for this filename extension with the command:

   Config(PG_E_SYNTAX,1,LOCAL)
3. In the "Syntax highlighting" section of **`startup.vdm`**, load the desired syntax definition file with the **Syntax_Load( )** command. For example, to load syntax highlighting for Pascal, use the command:

   Syntax_Load("pascal.syn")

   Be sure that any other **Syntax_Load( )** commands in this section are disabled. VEDIT can only load one syntax file at a time.

Only one syntax definition file can be loaded into VEDIT at a time. However, the following "trick" can load different syntax definition files for different projects:

1. Change the **Syntax_Load( )** command in the **`startup.vdm`** file to:

   Syntax_Load("proj.syn")
2. Copy the desired ".syn" file to "proj.syn" in each project's directory. VEDIT first searches the local directory for "proj.syn" before looking in the *VEDIT Home Directory*.

Notes:

Screen updating is slower with syntax highlighting enabled, but on Pentium computers, it is barely noticeable.

Creating your own ".SYN" syntax definition file

Custom syntax highlighting definition files are fairly easy to create for other languages and even non-programming applications. The on-line help topic "Color syntax highlighting" (DOS: "SYNHI") describes this in detail.

Most of the supplied ".syn" files were created by customers and then shared with us.

Template Editing

With each normal text character entered in Visual Mode, a "*template editing*" event macro can be executed. This requires that the macro is already loaded and that template editing is enabled.

The template editing macro typically performs some type of shorthand expansion by recognizing a key-word and expanding it to the full string of characters. For example, if the template macro file **`c.vtm`** is loaded, typing "**if (**" immediately expands to:

```
if ( ) {
  ...
}
```

The cursor is placed inside the "( )". It similarly expands other common C and VEDIT PLUS programming language statements.

If you accidentally type a key-word and get an undesired expansion, immediately select **{EDIT, Undo, Edit}** (**<Ctrl-Z>** or **<Alt-Bksp>**) to undo the expansion and your last typed character.

To insert the characters of a key-word without getting an expansion, precede the last character with **[ENTER CTRL]** (**<Ctrl-Q>**).

Manual Setup

You can initially try out template editing by loading a template editing macro from within VEDIT. However, if you regularly want to use template editing, you should enable it inside the **`startup.vdm`** file.

➢ **To setup template editing from within VEDIT:**

1. Select **{MISC, Load template file}**.
2. Select the desired ".vtm" file using Point and Shoot. **`vedit.vtm`** is a combination of **`c.vtm`** and **`html.vtm`** which only works in conjunction with the **`startup.vdm`** file.
3. Enable **{CONFIG, Programming, Enable template editing}** if necessary. **{MISC, Load template file}** automatically enables template editing in the current file.
4. If you loaded **`c.vtm`**, try typing "if (" to see the expansion. If you loaded **`html.vtm`**, try typing ".he" which should expand to "<HEAD><HEAD>" with the cursor in the middle.

Automatic template editing for C and HTML

The supplied **`startup.vdm`** file contains the commands to perform template editing (and color syntax highlighting) for C and HTML files. This is part of the "File-type specific configuration". When you open a file with a ".c", ".cpp" or ".h" filename extension, template editing for C is automatically enabled. When you open an .html, .htm or .htl file, template editing for HTML is

automatically enabled. Template editing is disabled for other files. This automatic template editing must be enabled.

➢ **To set up automatic template editing:**

1. Set **{CONFIG, Programming, File-type specific config}** to "3" or "7". A value of "7" also enables color syntax highlighting.
2. If desired, select **{CONFIG, Save config}** to ensure that the configuration change is permanent for the next time you run VEDIT. (This step in not needed if **{CONFIG, Auto-save config}** is enabled.)
3. Test the feature by opening a file with a ".c" extension. Check that **{CONFIG, Programming, Enable template editing}** is on.

   Open a file with a ".txt" extension. Check that **{CONFIG, Programming, Enable template editing}** is off.

-OR-

1. Edit the file **`startup.vdm`** in the *User Config Directory*.

   Search for "Config(PG_F_AUTO_CFG,1+2+4)" and delete the preceding "//" to enable the command.

   Config(PG_F_AUTO_CFG,1+2+4) //Enable file-type config

2. Save the file and exit VEDIT.
3. Restart VEDIT to process the new **`startup.vdm`** file. Check that **{CONFIG, Programming, File-type specific config}** is set to "7". Test the feature by opening a ".c" file.

Notes:

Template editing is closely related to the topic "Syntax Highlighting".

Only one template editing macro can be loaded into VEDIT at a time. However, **`vedit.vtm`** in conjunction with **`startup.vdm`** demonstrates how two or more languages can be supported in one macro.

See also:

The topic "Startup.vdm File".
The topic "File-type Specific Configuration".
The topic "Syntax Highlighting".
The topic "HTML Editing Features".

HTML Editing Features

VEDIT has several features that simplify editing HTML files, used for Internet Web pages.

- Template editing lets you enter simple two-letter codes which are automatically expanded to the full HTML codes. For example, typing ".he" expands to "<HEAD><\HEAD>" with the cursor in the middle.
- Enhanced keyboard layout with many hot-keys for common HTML codes. For example, pressing **<Alt-1>** inserts "<H1><\H1>" with the cursor in the middle.
- Color syntax highlighting for common HTML codes.
- The macro **`html2txt.vdm`** strips out all HTML codes to create a simple text file. The macro **`txt2html.vdm`** creates a simple HTML file from a text file.
- The **`.\user-mac`** sub-directory contains user supplied macros for website development. This includes **`WebXref.vdm`** which creates a cross reference of all files used in a website.

HTML files can be edited using template editing or hot-keys, or both at the same time. Template editing is performed by the macro files **`html.vtm`** or **`vedit.vtm`**. In addition, **`html.key`** is the "normal" keyboard layout with the HTML hot-keys.

VEDIT's HTML support is oriented towards experienced Web page creators (Webmasters) that want to create Web pages quickly with as few keystrokes and errors as possible.

The on-line help topic "HTML Editing Features" (DOS: "HTML") describes this in more detail.

Command Macros

"*Command macros*" are sequences of commands written in the VEDIT PLUS macro language. This topic covers everything the casual VEDIT user needs to know about command macros — primarily how to load and run the supplied macros. This topic assumes you are familiar with starting VEDIT and understand "text registers", both covered in Chapter 4 (Editing Guide).

VEDIT PLUS Macro Language

The VEDIT PLUS macro language is a complete text oriented programming language. It has arithmetic capabilities, numeric, character and string comparison, if-then-else decision making, looping, user input, screen output, window control and much more. Its over 200 commands can perform almost any conceivable character, line, block or file operation. The DOS version also has special commands for peeking/poking memory, accessing I/O ports and interfacing with low level DOS programming functions.

With VEDIT PLUS you can access the macro language at any time by pressing a key to enter "*Command Mode*". At the "COMMAND:" prompt, you can then simply enter one or more lines of macro language commands and VEDIT PLUS immediately executes the commands.

Besides the (technical) "COMMAND:" prompt, command macros can be executed in three ways:

- "Short" command macros can be executed as keystroke macros. A sequence of up to 1024 characters can be executed.

Many keys in the layouts to emulate Brief, Word Perfect and WordStar are implemented as such keystroke macros.

The supplied file **`key-mac.lib`** contains numerous keystroke macros consisting of macro language commands.

- "Longer" command macros are typically stored as disk files. These command macros can be loaded into the text registers and executed with **{MISC, Load/Execute macro}.**

Once a command macro is loaded into VEDIT, it can be executed with **{MISC, Execute macro}.**

A command macro can also be "auto-executed" when VEDIT starts up.

- Command macros can also be executed from the **{USER}** and **{TOOL}** menus. Short macros are typically included in the **`user.mnu`** file, longer macros are typically loaded from a file with the **Call_File( )** command.

Notes:

The main editing mode of VEDIT and VEDIT PLUS is called "*Visual Mode*". The distinction between "Command Mode" and "Visual Mode" is especially important to VEDIT PLUS users. VEDIT derived its name from "Visual EDITor".

The topic “Keystroke Macros” in Chapter 4 (Editing Guide) gives a step-by-step example of adding a keystroke macro listed in the **`key-mac.lib`** file with the **{CONFIG, Keystroke layout, Add keystroke macro}** function.

You may find it easier to add keystroke macros by editing the keyboard layout with **{CONFIG, Keyboard layout, Edit/view layout}**. The topic “Editing the Keyboard Layout - Adding a Keystroke Macro from KEY-MAC.LIB” in Chapter 4 gives a describes how to “copy and paste” a new keystroke macro from **`key-mac.lib`** directly into your keyboard layout.

Command Macros and Text Registers

Simple command macros can be built into VEDIT as keystroke macros. However, more complex command macros are stored as files, typically 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.

All VEDIT command macros are in “source code” format. There are no compiled macros. Therefore, all macros can easily be viewed. While experienced (and adventurous) VEDIT users could make small modifications to these macros, it requires the extensive documentation that comes with VEDIT PLUS to fully understand the macro language.

The VEDIT PLUS macro language has a C-like syntax and a free-form format — each line can contain just one command or many commands.

Command macro files are run by first loading them into a text register, and then executing the register. **{MISC, Load/Execute macro}** performs this operation. Alternatively, if the macro is already loaded, **{MISC, Execute macro}** executes it directly.

These functions load/execute the macro in text register “100” by default. Although some macros can execute from other registers, you should assume that macros are to be loaded into register “100 ”. (Theoretically, you could load multiple macros into different text registers, but that is a VEDIT PLUS topic.)

There is nothing special about the way command macros are stored in text registers — there is no difference between text registers that contain “cut and paste” blocks of text and those that contain command macros. To avoid confusion, we recommend using registers “ 0” through “9” for block operations. The remaining registers are then available for command 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.

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

Some macros must be executed from a particular text register, such as “100”, while others can be executed from any text register. (The beginning of a macro should document what registers it uses.)

This chapter assumes that all command macros are loaded into text register "100".

Notes:

Several functions are implemented as command macros that VEDIT automatically loads as needed. These macros are loaded into "hidden" text register 120, and therefore will not interfere with any macros that you are running.

Loading and Executing Command Macros

➢ **To load and execute a command macro:**

Many of our supplied macros can be run from the **{MISC, More macros}** menu. Others can be run by loading and executing them with **{MISC, Load/Execute macro}**.

➢ **To load and execute a command macro:**

1. Select **{MISC, Load/Execute macro}** (**<Ctrl-F7>**).
2. At the "Filename:" prompt, either enter the name of the macro, or use point & shoot file selection. The file selection dialog box defaults to the **.vdm** files in the *VEDIT Macro Directory*, e.g. **`c:\vedit\macros\*.vdm`**, for quick macro selection.
3. The dialog box also prompts with "Register number: ". Although most macros can be loaded and executed from any text register, others will only work from register 100. Therefore, it is best to simply select the default register "100 ".

NOTES: 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 VEDIT in order to exit the macro. Therefore, be sure to save your files before running unfamiliar macros.

Auto-Execution

VEDIT's *auto-execution* lets you specify a command macro to be run as soon as VEDIT starts up and has loaded any specified file(s) for editing.

-x *execfile* '*execfile*' is loaded into text register 100 and executed as a command macro. If no filename extension is given ".vdm" is assumed. '*execfile*' will be executed in addition to and after the **`startup.vdm`** file.

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

```
vpw -x print.vdm datafile.dat
```

VEDIT first loads and executes **`startup.vdm`** as usual. It then loads **`print.vdm`**. It then opens the file **`datafile.dat`**. Last, it executes the **`print.vdm`** macro. The auto-execution file is always loaded into text register 100 and executed from there.

When an auto-execution file is specified, VEDIT looks for it first in the current directory, then in the *User Macro Directory*, then in the *VEDIT Macro Directory*, and last in the *VEDIT Home Directory*.

You can also specify a few macro commands to be executed on the invocation line with the "**-c**" startup option.

-c *command* The macro language commands '*command*' are executed upon startup. '*command*' may be delimited with quotation marks ("); otherwise it ends on the first space.

+c *command* Same as "**-c**", except that the commands are executed *before* the **`startup.vdm`** file.

For example, the command to start up the editor and position the cursor at the first occurrence of the string "error" in the file **`datafile.dat`** is:

```
vpw -cSearch(/error/) datafile.dat
```

-OR-

```
vpw -c"Search(/error/)" datafile.dat
```

When using the "**-c**" and "**-x**" options, the order in which options are specified can become important. This is especially true when the "VEDIT" environment variable is used to specify default options. VEDIT first processes the "VEDIT" environment options from left to right; then it processes any command line options from left to right. The **`startup.vdm`** file is processed after any "+" options, but before the first "-" option.

Auto-execution macros can be specified in the Windows version by changing the "Target" in the icon's properties, or by using the "Run" command. See "Starting VEDIT for Windows" in Chapter 4 for more information.

PRINT - Print Macro

The PRINT.VDM macro can be selected as an option in the **{FILE, Print}** dialog box. It adds 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.

`print.vdm` documents how to optionally print line numbers and/or file positions on the left side of the page. Only trivial editing changes are needed.

➢ **To print a file with PRINT.VDM:**

1. Select **{FILE, Print}**.
2. In the print dialog box select "**( ) PRINT.VDM macro**". Then select **[Ok]**.

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

Alternatively, PRINT.VDM can be *auto-executed* when VEDIT is invoked. This is easily done from a (DOS) command line. In the Windows version, you can also create a special VEDIT icon that starts up with the PRINT.VDM macro.

➢ **To print a file with PRINT.VDM from a command line:**

1. Give the command:

   vpw -x print.vdm *filename*

   (VEDIT PLUS for Windows 3.1 is "**veditpw**". VEDIT PLUS for DOS is "**vedit**".)

   The entire file should begin printing. To stop the printing before it is done press <**Ctrl-C**>) or <**Ctrl-Break**>).
2. When done, PRINT.VDM gives you the choice of printing another file or returning to the operating system (OS).

NOTE: PRINT.VDM is intended as a macro example which is relatively easy to understand and enhance. Much more sophisticated formatters can be written in the VEDIT PLUS macro language. If they are named "print.vdm", they can easily be accessed from the print dialog box.

See also:

On-line help for **{FILE, Print}** dialog box.

WILDFILE - Multi-file Processing

WILDFILE.VDM is probably the most useful macro supplied with VEDIT. It 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 "*****". These files will be searched in the current directory and, optionally, in all subdirectories.

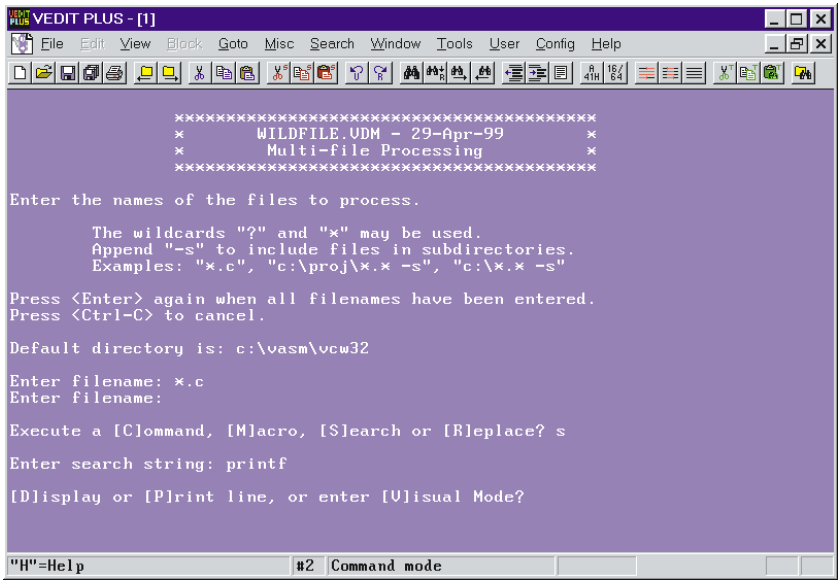
Since VEDIT can edit ***any*** file, including binary files such as ".EXE" executables, you can search through all files in a directory by specifying ***.*** without worrying about what kind of files they are. ***.*** will also search any "hidden" files.

The primary use of WILDFILE.VDM is to search for all occurrences of a word (variable name, etc.) in a large group of files. For example, you might want to view all occurrences of the word "printf" in all the **.C** files.

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.

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

1. Select **{MISC, WILDFILE macro}.**



2. At the filename prompt, enter "***.c**" and press **<Enter>** twice. This will search for all **.c** files in the current directory; if necessary, enter the full pathname.

You could search all **.c** files on the entire drive C: by entering "**c:*.c -s**".

3. At the next prompt, type "**S**" to select [S]earch. At the prompt for the search string, enter "printf".
4. At the next prompt, type "**V**" to view each occurrence in Visual Mode.

WILDFILE can also perform global replacements on many files. For example, you might have misspelled "parallel" as "parralel" in a group of **.TXT** files.

➢ **To correct a spelling error in all *.TXT files:**

1. Select **{MISC, WILDFILE macro}.**
2. At the filename prompt, enter "***.txt**" and press **<Enter>** twice.
3. At the next prompt, type "**R**" to select [R]eplace. 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.

Auto-executing the WILDFILE.VDM macro

The Windows version installation creates the "VEDIT Wildfile Macro" icon. It starts up VEDIT and immediately executes the WILDFILE.VDM macro.

When WILDFILE is done, it gives you the choice of running it again, exiting the macro but staying in VEDIT, or completely exiting VEDIT.

The DOS version of VEDIT can run the WILDFILE macro by auto-execution or with the supplied **`wild.bat`** file:

vedit -x wildfile.vdm

-OR-

wild

The supplied **`wild.bat`** file contains the command "vedit -x wildfile.vdm". To use it, **`wild.bat`** must be in the current directory or in one of the directories specified by the "PATH=" command in your **`autoexec.bat`** file.

Notes:

After selecting **{MISC, WILDFILE macro}**, the files specified for processing can include any files that are already open in VEDIT. This way you can perform an operation, such as searching through many files, without having to worry about which files are currently open for editing.

All files on the entire drive can be processed with "**c:*.* -s**". (It may take several minutes to process this command.)

COMPARE - Compare Files

The COMPARE.VDM macro can compare two text files of arbitrary size. You can edit the files as you are comparing them and copy blocks of text between them. It is ideal for merging the work done by several people on the same file(s), or determining the differences between two versions of a file. (The programmers that develop VEDIT merge their work this way.)

Unlike the **{SEARCH, Compare buffers}** function, COMPARE.VDM can automatically re-align the "*active*" file with the "*template*" file when you resume the comparison.

You can start this macro via auto-execution or from within VEDIT:

vpw -x compare.vdm

-OR-

1. Select **{MISC, More macros, Compare}**.

If you started the COMPARE macro from within VEDIT and buffers 1 and 2 already have open files, it prompts whether you want to save, abandon or compare these files.

COMPARE then prompts for the window configuration you want. You can select a vertical split, a horizontal split, or full-sized overlapping windows. It then prompts for the names of the *active* and *template* files. Either enter the filenames or press **<Enter>** for point and shoot file selection.

COMPARE then switches to the active file and places the cursor at the position of the first difference. Assuming that you selected split windows, you will also see the cursor in the template file. If desired, you can edit either file as desired, perhaps copying blocks from one file to the other.

To continue the file comparison, switch to the active file (if needed) and position the cursor where the files are again identical for at least 24 characters. For example, you might press **[NEXT LINE]** (**<Ctrl-Enter>**). **You only need to position the cursor in the active file, not in the template file**. COMPARE will align the template file itself.

Then press **[VISUAL EXIT]** (**<Ctrl-E>**). The cursor will advance to the next difference. This process is continued until the end of one or both files is reached.

If desired, you can edit either file as desired, perhaps copying blocks from one file to the other. You can switch between the files in the normal manner or by pressing **<F12>**.

- Select **{FILE, Next buffer}** (**<F6>** or **{FILE, Previous buffer}** (**<F5>**.
- Assuming that you selected split windows, click the mouse in the desired window.

HINT: You can switch between the files by pressing **<F12>**. This is particularly handy if you are editing additional files and don't want to toggle between all of them.

COMPARE makes a temporary assignment to **<F12>** and removes it when the macro is done.

Pressing **[ESCAPE]** during the comparison brings up the following menu:

```
FILE COMPARISON INTERRUPTED! Select from following options:

   [1] Examine active file          3) Resume, no alignment
   [2] Examine template file        4) Realign template & resume
   [5] Stop. Get exit options menu
   Enter Option:
```

Options [1] and [2] switch to the desired file.

Option [4] is the same as pressing **[VISUAL EXIT]** from the active file. COMPARE examines the 24 characters following the cursor and attempts to match them in the template file; if this alignment is successful, it resumes the file comparison.

Option [3] immediately resumes the file comparison from the current cursor positions, without attempting any realignment. You can also select this option by **[VISUAL EXIT]** from the template file.

When COMPARE is done, it gives you the choice of running it again, exiting the macro but staying in VEDIT, or completely exiting VEDIT.

Notes:

When comparing identically named files on two drives, you only need to enter the drive name at the "Template file:" prompt. For example, to compare "newdoco.txt" in the current directory of default drive "C:" with the same filename in the current directory of drive "D:", you could enter:

```
Enter the name of the active file:   newdoco.txt
Enter the name of the template file: d:
```

The comparison will either be case sensitive or insensitive, depending upon the setting of **{CONFIG, Search options, Default case-sensitive option}**.

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 20 lines preceding the template cursor or within 100 lines following the cursor, it gives 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.

COMPDIR - Compare Directories

The COMPDIR.VDM macro quickly compares all files in two directories and displays which files are different and which are unique to each directory.

COMPDIR first checks each pair of files time/date stamp and size; if they are the same, it assumes that the files are the same. Otherwise, it compares the files byte-by-byte to check if they really are different.

COMPDIR.VDM is currently only designed to work with the DOS version of VEDIT. The COMPDIR.BAT batch file must be used to start the directory comparison; COMPDIR.VDM cannot be run from within VEDIT.

```
compdir \direc1 \direc2
```

When the comparison is done, the top window displays the names of the files that are different in the two directories. The middle windows display the filenames that are unique in each directory, i.e. files that are in one directory and not in the other. The bottom window gives a short description of each window.

If desired, you can print the contents of each window, by switching to the desired window and selecting **{FILE, Print}**.

To exit, select **{FILE, Exit}**, then select **[Quit-all]** and confirm with **[Ok]**.

SORT - Sorting Macro

The SORT.VDM macro alphabetically sorts records consisting of multiple lines. Each record can either consist of the same number of lines, or the records can be separated by a blank line. The "sort key" is simply the entire first line; specific columns cannot be selected.

NOTE: Simple lines and single-line records are better sorted with the **{BLOCK, Edit/Translate, Sort lines}** function, which lets you sort according to any specified columns (field).

A simple mailing list consisting of address lines separated by one or more blank lines can be sorted by SORT.VDM. 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

The SORT macro can be started from within VEDIT, or by auto-execution from the DOS prompt or the Windows "Run" command. However, you must start SORT from within VEDIT when editing files with fixed-length records so that you can set the record length before beginning the sort.

➢ **To sort a file already opened in VEDIT:**

1. If it is a database file with fixed-length records, be sure that **{CONFIG, File handling, File type}** and **{CONFIG, File handling, Record header size}** are set correctly.
2. By default, the sorting is not case sensitive. To make it case sensitive, enable **{CONFIG, Search options, Default case sensitive option}**.
3. Select **{MISC, More macros, Sort}**.

If no file is yet open, SORT prompts for the filename of the file to be sorted and for the name of the file to contain the sorted output.

SORT then prompts for the number of lines in each record; enter the number, or "0" if the records are separated by a blank line. The entire file will then be sorted.

Running SORT.VDM via Windows "Run" command or from DOS

The SORT macro can also be started from the DOS prompt, the Windows "Run" command or a Windows icon with the appropriate properties.

vpw -x sort.vdm

(VEDIT PLUS for Windows 3.1 is "**veditpw**". VEDIT PLUS for DOS is "**vedit**". VEDIT PLUS for QNX is "**vp**".)

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 they are the same.

You can also specify the name of the file to be sorted:

vpw -x sort.vdm *filename*

Alternatively, you can specify the (input) file to be sorted and the (output) file to contain the sorted output:

vpw -x sort.vdm *infile* **-a** *outfile*

SORT then prompts for the number of lines in each record; enter the number, or "0" if the records are separated by a blank line. The entire file will then be sorted.

Alternatively, you can specify the number of lines in each record with the "**-n**" option:

vpw -x sort.vdm -n4 *filename*

vpw -x sort.vdm -n0 *filename*

This format sorts a file without any prompts or user intervention.

Notes:

The SORT macro can realistically handle files up to a few megabytes. A typical 1 Megabyte file with several thousand records will be sorted in about 1 minute (200mhz Pentium). Multi-megabyte files may take unreasonably long to sort; however SORT displays its progress and can be interrupted at any time.

HINT: To sort multiple line records according to a specific field (columns), you may be able to convert the record into a single line, sort it with **{BLOCK, Edit/translate, Sort lines}** and then split each line back into the desired multiple lines.

DBASE.VDM Macro

The DBASE.VDM macro simplifies editing dBASE III type ".DBF" database files. It sets the correct record size and header offset for the dBase III data file in the current buffer, i.e. it automatically sets **{CONFIG, File handling, File type}** and **{CONFIG, File handling, Record header size}**. This way, the records will be properly aligned on the screen and the "LINE:" display on the status line will display the correct record number.

DBASE.VDM also sets up table information in an unused buffer. This lets you view the field names, field types, field sizes and determine which column each field begins in. For example, it can create a display such as the following:

```
Last updated: 2/23/94
Total Records: 20

Record Length: 182
Header Length: 449

Field  Field Name  Type        COL:  Width   Dec.
=================================================
   1  CO          Logical       2       1
   2  LN_CO       Logical       3       1
   3  LBL_LINES   Numeric       4       1      0
   4  HOW_WIDE    Numeric       5       2      0
   5  TO_LINE     Character     7      32
   6  ADDR1       Character    39      32
   7  ADDR2       Character    71      32
   8  ADDR3       Character   103      32
   9  CITY        Character   135      18
  10  STATE       Character   153       2
  11  ZIP         Character   155      10
  12  ADD_DATE    Date        165       8
  13  NOTES       Memo        173      10
```

- **To run DBASE.VDM from within VEDIT:**

  1. Switch to the buffer containing the .DBF file.
  2. Select **{MISC, More macros, Dbase}**.

  The .DBF file should now be correctly displayed. You can check the "record size" and "record header size" in **{CONFIG, File handling}**.

  To view the table information, select **{FILE, Next buffer}** (default: **<F6>**) until you see it.

Optional “Hot-key” for xBase Files

You can quickly configure VEDIT to an xBase file by selecting **{MISC, More macros, Dbasekey}**. This sets the correct “record size” and “record header size”. All fields will then immediately line up on the screen. However, this simpler macro does not create the detailed table information display shown above.

If you often use this function, you may want to set up a hot-key for it. You can either add the hot-key to your **`vedit.key`** file or set it up in the **`startup.vdm`**. The default hot-key is **<Alt-F12>**.

➢ **To enable the xBase hot-key in `startup.vdm`:**

1. Open the file **`startup.vdm`** in the *User Config Directory* for editing.
2. Locate the line: // Key_Add(“Alt-F12"...
3. Delete the leftmost “//” to activate the command.
4. Save the file and exit VEDIT.
5. Restart VEDIT. Open an xBase file and press **<Alt-F12>** to confirm that it works.

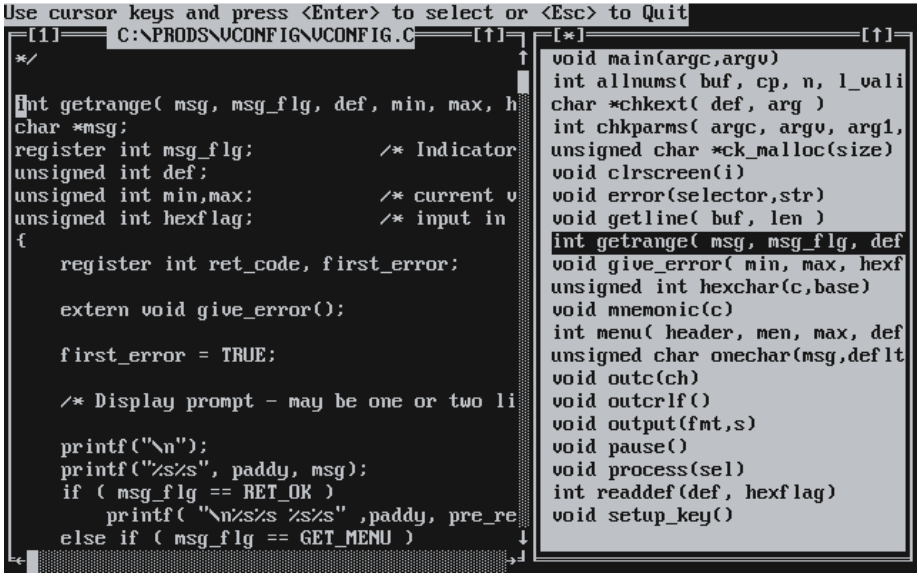
CFUNC - C Program Outliner

The CFUNC macro is a split-screen outliner that lists each C program routine declaration in a separate window; as you move through the list, the original window moves through the C program.

➢ **To start the C program outliner:**

1. Select **{MISC, More macros, Cfunc}**.

A typical screen display while the CFUNC.VDM macro is running is:



Use **<Cursor Up>** and **<Cursor Down>** to move through the outline. Then press **<Enter>** to resume editing at the current location in the outline.

After the macro has been loaded and used, you can easily restart it by selecting **{MISC, Execute macro}** (default: **<Ctrl-F8>**). At the prompt, simply press **<Enter>** to select register 100.

NOTES: PFUNC.VDM is the same macro for Pascal.

{USER, Search all and select} performs a similar function for any search string.

RUNSHELL - Run Other Programs

The function **{MISC, Run program}** is only suitable for running other programs (or DOS commands) that do not access the files currently open in VEDIT.

The functions **{USER, Save and run program 1}** and **{USER, Save and run program 2}** are designed to run compilers and other programs that need to access the currently open files.

These functions first prompt for the command to run a program (e.g. compiler). Enter the full command, including any parameters and options. The command is saved as the default command for the next time. After all open files are saved and closed, the command is executed by shelling out, possibly via a DOS box. When the program is done, any DOS box is auto-closed and all files are reopened.

To run a Windows program, precede the command with "**win:**". Depending upon which version of VEDIT is running (Win32, Win16 or DOS), and the operating system (WinNT, Win95/98, Win31, DOS), it may bypass the DOS box in order to run a Windows program.

NOTES: The program can open and even change the files you were editing in VEDIT.

The command to run a Windows program should normally be preceded with "**win:** ". Otherwise, VEDIT will continue running and will immediately re-open the files and lock them, preventing the program from accessing them.

If a Windows program is not running properly under Windows 95/98, you can try preceding the command with "**start /w**" instead of "**win:**".

The full pathname of the currently edited file can be passed to the program by including "##" as a parameter in the command. To illustrate this and how to run a Windows program, open a small text file, select **{USER, Save and run program 2}** and enter the command:

win: notepad ##

This will open the current text file in Notepad for editing. When you close Notepad, the file will be reopened in VEDIT and you should be able to see any changes made in Notepad.

If the program is set up to create the special file "**`vout`**", this file is automatically opened in a new window. For example, this can be used to view the output (error) messages from a compiler.

To illustrate the "**`vout`**" file, select **{USER, Save and run program 1}** and enter the command:

```
dir > vout
```

This redirects the directory command into the file "**vout**" which is then automatically opened in VEDIT. Before shelling out, any existing "**vout**" file in the current directory is deleted. In the unlikely event this is a problem, a trivial change documented in the **runshell.vdm** file disables the "vout" feature.

The **{USER, Save and run program 1}** and **{USER, Save and run program 2}** functions are implemented by the **runshell.vdm** macro. The only difference between these functions is the "slot" in which the command is saved. The last command is the default command the next time the function is selected. This makes it easy to run the same compiler command over and over again, or to change it as needed.

runshell.vdm can actually run and save up to five different commands. If desired, you can add a 3rd, 4th and 5th command to the **user.mnu** file by adding the following lines to it:

```
22
Save and run program 3
#103=3 CallF(122,"runshell.vdm")
22
Save and run program 4
#103=4 CallF(122,"runshell.vdm")
22
Save and run program 5
#103=5 CallF(122,"runshell.vdm")
```

If you always use the same commands, you can change the "Save and run program..." messages in the **user.mnu** file to something more descriptive. Users familiar with the VEDIT PLUS macro language can also change the **runshell.vdm** macro to better serve specific needs.

"ctags" Symbol Lookup

The "ctags" facility is useful to programmers that are working on large programs, particularly programs that consist of many files. Once setup, you can place the cursor on any function (subroutine) or symbol name and select **{MISC, More macros, Utags}** or press a hot-key (default: **<F12>**) to lookup the symbol. VEDIT will open the file in which the symbol is declared, with the cursor on the symbol's declaration. You can then press the hot-key again to return to the original file. (Other lookup options can be selected within the **utags.vdm** macro).

The "ctags" facility consists of two macro files — **ctags.vdm** and **utags.vdm**. **ctags.vdm** creates the **tags** database file with symbol declarations. **utags.vdm** performs the lookup function. The lookup can be performed by selecting **{MISC, More macros, Utags}**, but as a convenience, it is usually assigned to a hot-key by the **startup.vdm** file.

To set up for ctags and create the **tags** database, select **{MISC, More macros, Ctags}**. As supplied, **ctags.vdm** supports C and Assembly language. You can either create a new **tags** database or append to the existing one. It supports programs consisting of C and assembly language modules.

ctags.vdm also supports a user specified symbol "search" string for other languages. It could also be useful for non-programming applications.

ctags.vdm works similar to the WILDFILE macro. At the filename prompt, you enter a wildcard specification such as "***.c**", "***.h**" and "***.asm**". It then processes all specified files in the current directory. By adding "**-s**" to the filename, e.g. "***.c -s**", it processes all matching files in all subdirectories.

VEDIT's ctags facility therefore supports the biggest projects, even those with thousands of files in many subdirectories.

The format of the generated **tags** file is identical to that produced by the UNIX "ctags" utility. Therefore, as an alternative to the **ctags.vdm** macro, you should be able to produce the **tags** file with any utility that is "ctags" compatible.

Setup

To use ctags, you must create the **tags** database in the main project directory. (This is the directory you use to edit and compile the program.)

➢ **To create a "tags" database:**

1. Start VEDIT and open just one file in the main project directory. This will set VEDIT' "current" directory so that the **tags** database is created in this directory.
2. Run the **ctags.vdm** macro by selecting **{MISC, More macros, Ctags}**.

-OR-

1. At a (DOS/QNX/UNIX) command prompt, switch to the main project directory. The file **tags** will be created in this directory.

2. Run the **ctags.vdm** macro:

   vpw -x ctags

   (VEDIT PLUS for Windows 3.1 is "**veditpw**". VEDIT PLUS for DOS is "**vedit**".)

3. At the ctags macro's language prompt, select whether you are working in C or Assembler, or want to enter a custom symbol recognition search pattern.

4. At the filename prompt, enter a file specification, typically using the wildcard characters "*" and "?". E.g., enter "*.c" or "*.asm".

   Follow the filename with "-s" to process all matching files in all subdirectories. For example, "***.c -s**" will process all .C files in the current directory and all subdirectories.

   Enter as many file specifications as needed; if the files are not in the current directory, enter the full pathname.

   When all files have been entered, immediately press **<Enter>** again.

5. The ctags macro will then display "Processing <filename>" for each file processed. (Processing is quite fast - one Megabyte of source code will be processed in under one minute.)

You can perform a lookup by selecting **{MISC, More macros, Utags}**, but it is usually more convenient to set up a hot-key to perform the lookup. This is easily done in the **startup.vdm** (macro) file. The supplied **startup.vdm** file contains the necessary command. (However, the command has been disabled by preceding it with "// " — the comment characters.)

For example, to assign the lookup function to **<F12>**, add the following line to your **startup.vdm** file:

```
Key_Add("F12",'[MENU]MMU',OK)
```

Usage

Once setup, the ctags facility is trivial to use. Simply place the cursor on a function name, press the hot-key (default: **<F12>**) and VEDIT will switch to the function's declaration. Then press the hot-key again to return to the original file and position.

Advanced Usage Notes

You can easily modify **utags.vdm** to select what it does when the function declaration is found — simply switch to the file, switch to the file and cascade the windows, or (default) switch to the file and setup to switch back. View the **utags.vdm** file for more information. This macro is not overly complex and can be modified to your preferences.

utags.vdm only looks for the **tags** file in the current directory. If necessary, you could copy **utags.vdm** to various local directories and modify the "**File_Open( )**" command to specify the full pathname to the **tags** file.

(We would appreciate any improvements you might make to these macros.)

Integrated Compiler Support

Overview

VEDIT's compiler support can improve a programmer's productivity by allowing any desired compiler, assembler, linker, debugger and Make utilities to be run from within the editor.

The compiler support is implemented as nine items in the **{Tools}** menu. If there is no **{Tools}** menu, e.g. the **{Tutorial}** menu is displayed, the correct **{Tools}** menu can be loaded by selecting **{MISC, Load Compiler support}**.

Either the "normal" compiler support can be loaded into the **{Tools}** menu or Java SDK specific support can be loaded into the **{JavaTools}** menu. You can easily switch between Java SDK and another compiler by selecting **{MISC, Load compiler support}** again.

The item **{TOOLS, Compile}** runs the currently selected compiler. If an error is reported, the correct source code file is loaded and the cursor placed on the line containing the error; the entire error reported by the compiler is displayed in a separate window. You can immediately edit the file and with a hot-key advance to the next error or re-compile the program.

Once your program compiles without errors you can also run your linker and debugger from within VEDIT.

The normal compiler support also supports several popular Make utilities and will even track errors reported by different compilers in a Make script.

The compiler support consists of the **`compile.vdm`** macro which loads either **`compile.mnu`** or **`java-sdk.mnu`** as the new **{Tools}** menu, the **`c-xxxxxx.vdm`** macros which implement each **{Tools}** menu function, compiler specific ".vcs" files, and the **`compile.cnf`** and **`java-sdk.cnf`** configuration files.

For each supported compiler there is a corresponding ".vcs" macro file. The file names correspond to the compiler's command name, e.g. the Microsoft C compiler is **`cl.vcs`**, the Borland C compiler is **`bcc.vcs`**, and the Java SDK compiler is **`javac.vcs`**.

The **`compile.vdm`**, **`compile.mnu`**, **`java.mnu`**, **`compile.cnf`** and **`java-sdk.cnf`** files are located in the *VEDIT Home Directory*, the remaining files are in the \COMPILE subdirectory, typically "c:\vedit\compile".

The **`compile.cnf`** file determines the default Compile, Link, Debug and Make commands that will be used. You must edit this file to set the default commands and compiler options. Optionally, you can override the **`compile.cnf`** file by creating a **`compile.vco`** file in a project's source code directory which specifies the Compile and other commands to be used for that project. For example, you may have a program in one directory which is compiled as a 32-bit program, and a program in another directory which is compiled as a 16-bit program.

Similarly, the **`java-sdk.cnf`** file determines the default Compile, Compile-with-Debug, Debug and Java-VM commands that will be used when Java SDK has been selected.

The fully commented sample macro **`sample.vcs`** is an excellent guide for creating your own compiler specific macro. (Of course, we would appreciate receiving any compiler specific macros that we can add to VEDIT.)

Compiler Support Installation

The installation procedure for the Windows and DOS versions of VEDIT gave you the option of installing the compiler support files. If you did not initially select this option, you must reinstall either version of VEDIT. Since the installation can save your current configuration and keyboard layout, this is a simple procedure that only takes a few minutes.

Make sure that the compiler is functioning and that all necessary PATH and SET statements are performed prior to running the compiler support.

Enable Compiler Support

The compiler support is implemented as items in the **{Tools}** menu. However, by default the **{Tutorial}** menu is displayed in place of the **{Tools}** menu.

You can manually set up the **{Tools}** menu with the compiler support items by selecting **{MISC, Load Compiler support}**. Similarly, you can switch back to the **{Tutorial}** menu by selecting **{MISC, Load Tutorial menu}**.

Assuming you want the compiler support loaded as the default **{Tools}** menu, you should make a simple change to the **`startup.vdm`** file.

➢ **To enable compiler support as the default {Tools} menu:**

1. Edit the startup file **`startup.vdm`**.
2. Search for the string "#120 = 0" (without quotes) at the beginning of a line.

   Change it to "#120 = 1" for normal compiler support in the **{Tools}** menu.

   Or, change it to "#120 = 2" for Java SDK support in the **{JavaTools}** menu.
3. Restart VEDIT. The **{Tools}** or **{JavaTools}** menu should now display the compiler support items.

NOTES: Enabling the compiler support with **{MISC, Load Compiler support}** or in **`startup.vdm`** runs the macro **`compile.vdm`** which also changes your keyboard layout!

`compile.vdm` sets up the hot-keys for the common compiler support items in the **{Tools}** menu. These additional hot-keys are designed for the "Normal" keyboard layout, but might cause conflicts with other layouts. In this case, you must change the hot-keys by editing the **`compile.vdm`** file directly.

Configuring COMPILE.CNF (or JAVA-SDK.CNF)

The **`compile.cnf`** file specifies the default Compile, Link, Debug and Make commands to be used. Open this file for editing. You will see the following:

```
COMPILE.CNF - Default compiler support.
              Must be located in VEDIT Home Directory
              e.g. C:\VEDIT.
SUPPORTPATH=
DEFCOMPILE=CL /c -proj.ext
DEFLINK=LINK /codeview -proj.obj;
DEFDEBUG=CV /e -proj.exe
DEFMAKE=NMAKE /n

COLORDISPLAY=A // A=Auto; Y=Color; N=B/W; ?=Prompt
```

The following parameters must be set:

SUPPORTPATH

Specifies the path to the compiler support files, particularly the ".vcs" files. **It does not need to be set if the installed location is used**, i.e. the "\compile" subdirectory in the *VEDIT Home Directory*, e.g. "c:\vedit\compile".

DEFCOMPILE

This specifies the default Compiler command (e.g. "cl /c -proj.ext" or "bcc -c -proj.ext"). The name of the compiler must have an associated ".vcs" file (e.g. "cl.vcs" or "bcc.vcs").

Typically, a Compiler command will consist of the compiler name, any desired compiler options and "-proj.ext". "-proj.ext" specifies where in the compile command the name of the (project) file being compiled should appear.

When running the compiler, "-proj.ext" will be expanded to the full project filename and extension. Alternatively, "-proj" will be expanded to the project filename without extension.

"-proj" will not interfere with any "-p" option your compiler may have.

You should set DEFCOMPILE to your most commonly used Compiler command.

DEFLINK, DEFDEBUG, DEFMAKE

These specify the default Linker, Debugger and Make commands. There are no associated .vcs files.

The Linker command will typically include "-proj" to specify where the name of the project file should appear. "-proj" is immediately followed by the filename extension of the object files, typically ".obj" or ".o".

If you have trouble with the Microsoft "NMAKE", try adding the "/n" option. Similarly, with the Borland or Avocet "MAKE", try adding the "-n" option.

Java SDK: In the **`java-sdk.cnf`** file, DEFLINK actually sets the "Compile with debug" command and DEFMAKE sets the "Java VM" command.

COLORDISPLAY, WINDOW ATTRIBUTES...

These specify the colors used by the compiler support. They are documented in the **`compile.cnf`** file.

After editing **`compile.cnf`**, the compiler support should now be properly installed and functional.

- **Example COMPILE.CNF for Microsoft C**

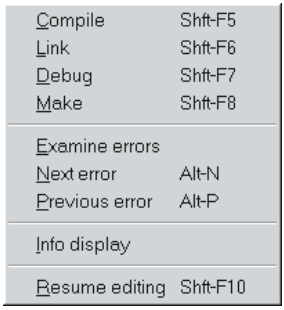
```
SUPPORTPATH=
DEFCOMPILE=cl /am /c -proj.ext
DEFLINK=link /codeview -proj.obj;
DEFDEBUG=cv /e -proj.exe
DEFMAKE=nmake
```

- **Example COMPILE.CNF for Borland Turbo Assembler**

```
SUPPORTPATH=
DEFCOMPILE=tasm /zi -proj.ext
DEFLINK=tlink /m/v -proj.obj
DEFDEBUG=td -proj.exe
DEFMAKE=make
```

Running the Compiler Support

Assuming you have enabled the compiler support and have configured the **`compile.cnf`** (or **`java-sdk.cnf`**) file correctly, you are ready to use the compiler support.



- **To run the compiler support:**

1. Start VEDIT and open the main file to be compiled. E.g., you want to be switched to the main ".c" file and not to a ".h" or other "include" file.

   The current file will be the "project file". This is the filename that is given to the compiler or linker command. (Via the "-proj.ext" expansion described above.)

If you are using a "Make" file to compile and link your project, you do not need be switched to the main project file.

2. Select **{TOOLS, Compile}** or press its hot-key. The compiler will immediately run; you may temporarily see some output on the screen.

   Note: If you do not have a **{TOOLS, Compile}** item, refer to the section "Enable compiler support" above.

3. When the compiler is done, the output from the compiler will be displayed in a VEDIT window.

   If there were no compilation errors, or you don't want to look at them now, simply press **<Enter>** or **<Esc>** to resume editing.

   If there were errors and you want to go to the first one now, press "**F**". It will switch to the source code file with the error and position the curson on the line containing the error. The full error message will be displayed in a separate window.

   To browse the compiler output and optionally select an error, press "**B**".

4. Select **{TOOLS, Next error}** (hot-key: **<Alt-N>**), **{TOOLS, Previous error}** (hot-key: **<Alt-P>**) to move the cursor to each source code line containing an error.

   Notes: These functions go to absolute line numbers. If you add or delete lines in the source file, they will go to the wrong line. If this becomes too confusing, you will probably want to re-compile.

   To remove the error window, either select **{TOOLS, Next/previous error}** until you reach the end, or select **{TOOLS, Resume editing}**.

5. When all errors are (hopefully) corrected, select **{TOOLS, Compile}** again to re-compile.

6. After a successful compilation, you can select **{TOOLS, Link}** to run the linker and then **{TOOLS, Debug}** to run the debugger.

   If desired, you can set up "Debug" to simply run the program without a debugger.

See also:

On-line help for compiler support items.

The on-line help topic "Compiler Support" (DOS: "COMPILE") contains complete details on using the compiler support.

You may want to print this topic from the Windows on-line help system.

To print this topic in the DOS version, open the on-line help file **`vhelp.hlp`** (VEDIT PLUS: **`vphelp.hlp`**). Since it is a normal text file, you can easily print portions of it. Search for "\compiler\". Highlight the entire topic and select **{FILE, Print}** to print it.

Chapter 6

Menu Reference

This chapter is a detailed description of the VEDIT menus and provides a quick reference for specific menu operations.

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 "Normal" keyboard layout which may be used to directly access each menu and many menu items.

The description of each menu item is divided into the following sections:

Menu Item — This is the name of the menu item as it appears on the screen.

Brief Description — Read this brief description to determine if you have found the desired item.

Keystroke Equivalent: — If the "normal" IBM PC keyboard layout has a "hot-key" or other keystroke equivalent for selecting this item, it is listed here. Many keystroke equivalents are set up as "keystroke macros"; other menu items are equivalent to basic edit functions. Use **{CONFIG, Keyboard layout, Edit/view layout}** to view all assigned keys and keystroke macros. For those items without keystroke equivalents, you can always create a new keystroke macro.

Full Description: — A detailed 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.

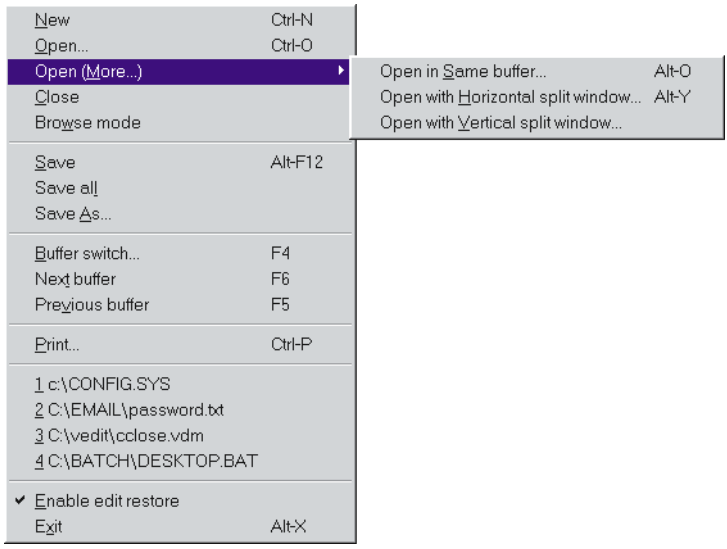
File Menu

Keystroke Equivalent:

<Alt-F>, This is a keystroke macro.

Introduction:

The File menu includes functions for opening a file for editing, switching between the multiple files being edited, saving and exiting.



New

Open a new edit buffer without an assigned filename.

Keystroke Equivalent:

<Ctrl-N>, This is a keystroke macro.

Full Description:

"New" opens a new (unused) edit buffer; it will initially be empty and have no assigned filename. To assign a filename to the new edit buffer, select **{FILE, Save as}**.

Experienced users will probably prefer to use **{FILE, Open}** even when creating new files.

Notes:

"New" is rarely used in VEDIT; it is provided for compatibility with other editors and word processors. Some other editors force you to select "New" and then "Save as" to create a new file. However, VEDIT automatically creates a new file when the file you select with **{FILE, Open}** doesn't already exist.

The buffer selected by "New" will be the lowest numbered buffer between #2 and #32 that is currently unused. Buffers that are currently open, but are empty, are not selected.

You can also open an empty edit buffer with **{FILE, Buffer switch}**. When the specified buffer is not already open, it is opened as an empty buffer.

VEDIT can have multiple "<Untitled>" buffers open without assigned file-names. However, selecting **{FILE, Exit}** and then "Save-all" does not save those edit buffers which have no filename.

See Also:

{FILE, Open}, **{FILE, Buffer switch}**

Open

Open an additional file for editing. If the selected file does not exist, it is created.

Keystroke Equivalent:

<Ctrl-O>, Keystroke macro for **{FILE, Open, New buffer}**.

Full Description:

"Open" opens (or creates) an additional file in an unused buffer and displays it in its own window.

The File-open dialog box optionally lets you open the file in "Read-only" mode so that you don't accidentally alter it and can navigate it more quickly. You can also open binary and fixed-length record files. The on-line help for the File-open dialog describes these options in detail.

➢ **To simultaneously edit another file in a new window:**

1. Select **{FILE, Open}** (**<Ctrl-O>**).
2. You are prompted for the name of the file to edit.

If VEDIT does not find a file with the specified name, it will create the file.

The new window will either be full-sized or cascaded, depending upon the setting of **{CONFIG, Display options, Auto-create window style}**.

The status line will display the new buffer number; the window border will display the new filename.

Notes:

VEDIT requires enough disk space to accommodate approximately 2 times the actual file size. You should always work with at least that amount of disk space available.

If you select a file which is already open in another edit buffer, "Open" only switches to that buffer; you cannot have the same file open in two buffers at once. To display different parts of one file in two windows, select **{WINDOW, Split}**.

The new buffer selected by "Open" will be the lowest numbered buffer between #1 and #32 that is currently unused. Buffers that are currently open, but are empty, are not selected.

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, Open}**.

See Also:

"Editing Multiple Files" in Chapter 4.
{FILE, New}, **{FILE, Open (More)}**
{WINDOW, Split}, **{WINDOW, Switch}**,
{VIEW, Zoom}, **{VIEW, Full size}**

Open More (Sub-menu)

VEDIT has three variations of **{FILE, Open}** to save you steps when editing multiple files.

Keystroke Equivalent:

<Alt-O>, Keystroke macro for **{FILE, Open, Same buffer}**.
<Alt-Y>, Keystroke macro for **{FILE, Open, Horizontal window}**.

Full Description:

{FILE, Open (More), Same buffer} (**<Alt-O>**) is a shortcut way of closing the current file and opening the next file. It saves keystrokes and/or mouse clicks. It is the best way of editing one file after another; it reduces the confusion of having many unnecessary files and windows open at once.

"Open with Horizontal split window" and "Open with Vertical split window" simplify opening two files side-by-side or one above the other, perhaps to compare them.

- **To edit a new file (in the same buffer):**

  1. Select **{FILE, Open (More), Same buffer}** (**<Alt-O>**).
  2. If the current buffer contains a modified file, you will be prompted whether the file should be saved or abandoned before opening the new one:

     ```
     Save current file? [Yes] [No-Abandon]
     ```

     Answer **[Yes]** to save the current file with all modifications. Answer **[No]** to abandon any changes made since the last time the file was saved.
  3. You are prompted for the name of the file to edit with the normal File-open dialog box.

- **To simultaneously edit two files in a split window:**

  1. Open the first file with **{FILE, Open}** (**<Ctrl-O>**) or **{FILE, Open (More), Same buffer}** (**<Alt-O>**).
  2. You probably will want to expand the window to full size, if not already, with **{VIEW, Full size}**.

3. Select **{FILE, Open (More), Horizontal window}** (**<Alt-Y>**) to display the new file in the lower half of the current window. Or select **{FILE, Open (More), Vertical window}** to display the new file in the right half of the current window.

4. You are prompted for the name of the file to edit.

   The current window will be split into two windows and you will now be editing the new file in the new window.

Notes:

Since you can later resize, tile, cascade or reset the windows, it is not critical in which window configuration you open the file.

See the Notes for **{FILE, Open}**.

See Also:

"Editing Multiple Files" in Chapter 4.

Close

Close the current edit buffer; save or abandon any current file.

Keystroke Equivalent:

(None) Use **{WINDOW, Close}** (**<Ctrl-F4>**) instead.

Full Description:

"Close" closes the current edit buffer and all corresponding (attached) windows. If the edit buffer contains modified text that has not been saved, you are prompted whether you want to save or abandon it.

When closing the last edit buffer, VEDIT keeps the buffer open as an empty buffer. VEDIT always keeps at least one buffer and window open.

NOTE: It is usually more convenient to use **{WINDOW, Close}** (**<Ctrl-F4>**) or to click the mouse on the window's "close" button to close both the window and file. When the buffer (file) is displayed in only one window, **{FILE, Close}** and **{WINDOW, Close}** are identical. There is only a difference when the buffer is displayed in two or more windows, e.g. after **{WINDOW, Split}**.

➢ **To close the current edit buffer and all attached windows:**

1. Select **{FILE, Close}**. If the buffer contains modified text, it prompts whether the current file should be saved or abandoned:

   ```
   Save current file? [Yes] [No-Abandon]
   ```

2. Answer **[Yes]** to save the current file. Answer **[No]** to abandon any changes that have been made since the last time the file was saved.

3. If you answer **[Yes]** to save the text, but no filename is assigned, you will be prompted for the "Save as" filename in which to save the file.

Notes:

VEDIT skips confirmation prompts when you abandon unmodified files and does not re-write unmodified files to disk.

Use **{WINDOW, Close}** (**<Ctrl-F4>**) or click the mouse on the window's "close" button to close both the window and the edit buffer. However, since the last window cannot be closed, **{WINDOW, Close}** is equivalent to **{FILE, Close}** when only one window exists.

In most cases **{WINDOW, Close}** is preferred over **{FILE, Close}**.

See Also:

"Editing Multiple Files - Closing Files" in Chapter 4.
{FILE, Open}, **{WINDOW, Close}**

Browse Mode

Enable/disable browse mode for the current file and the next file edited.

Keystroke Equivalent: (None)

Full Description:

When set, the current file cannot be altered any further. If VEDIT was invoked with the "**-B**" invocation option, or if the current file was opened in Read-only mode, this toggle is set and cannot be turned off.

This item is normally disabled so that the current file can be altered. When enabled, the current file cannot be altered, or if the file has already been altered, it cannot be altered any further.

CD-ROM and other read-only files are automatically opened in Read-only mode.

Notes:

Using browse mode is convenient when you want to ensure that a file is not accidentally altered.

See Also:

"Starting VEDIT - Read-only Mode" in Chapter 4.

Save

Save All

Save the current file or files and continue editing.

Keystroke Equivalent:

<Alt-F12> is a keystroke macro for "Save".

Full Description:

"Save" saves any changes you have made to the current file so that they won't be lost by power failure, system crash or a major editing mistake.

"Save all" similarly saves any changes you have made in all currently open files.

You should select "Save" or "Save all" frequently. It is important and takes very little time.

Alternatively, you can enable VEDIT's auto-save feature to save all files at regular intervals. See **{CONFIG, File handling, Auto-save interval}**.

➢ **To save the current file and continue editing:**

1. Select **{FILE, Save}**. If you made any changes since the last time you saved the file, the file will be saved to disk. If you did not make any changes, VEDIT will not bother to save the file again.
2. Continue editing the file. Remember to regularly save your changes.

Notes:

When editing multiple files, "Save" saves only the current file; this is useful when the other files are not in a state in which you want to save them. Otherwise, use "Save all" to save all modified files.

If the current buffer has no assigned filename, you will be prompted to enter the "Save as" filename.

"Save all" only saves text in the current edit buffer, in the main edit buffer #1 and in those buffers that have assigned filenames. If the current or the main edit buffer do not have filenames assigned, it will prompt for the "Save as" filename.

If **{CONFIG, File handling, Auto-save}** is enabled, VEDIT automatically performs a **{FILE, Save all}** at regular intervals.

See Also:

"Exiting VEDIT - Auto-file Save" in Chapter 4.
{FILE, Save all}, **{FILE, Save as}**, **{FILE, Exit}**

Save As

Name a file and save contents of buffer.

Keystroke Equivalent: (None)

Full Description:

"Save as" saves the file you are currently editing under a new name. This is convenient when you want to use an existing file as a template or prototype for a new file.

"Save As" also lets you save the contents of an edit buffer that currently has no filename associated with it.

➢ **To save the current file under a new name:**

1. Select **{FILE, Save as}**.
2. You are prompted for the desired filename. You could select an existing file, but more likely you will enter a new filename.

Notes:

You can save steps by entering the "save as" filename when you open the original file. At the filename prompt enter e.g. "oldfile.txt -a newfile.txt".

See Also:

{FILE, Save}, **{FILE, Exit}**,

Buffer Switch

Change to a different edit buffer and file.

Keystroke Equivalent:

<F4>, This is a keystroke macro.

Full Description:

"Buffer switch" switches directly to any desired edit buffer. If the specified buffer is already open, this is similar to using **{FILE, Next buffer}** or **{FILE, Previous buffer}**. If the buffer is not open (i.e. "unused"), it opens the buffer as an empty buffer, similar to **{FILE, New}**.

When simultaneously editing multiple files, **{FILE, Buffer switch}** can be used to switch directly to a given file. If that file is currently displayed in a different window, it also switches to that window.

➢ **To switch to a desired edit buffer (file):**

1. Select **{FILE, Buffer switch}** (**<F4>**).
2. In the dialog box you can either directly enter the desired buffer number or select a buffer using point & shoot.

The new edit buffer's number will be displayed on the status line.

Notes:

Selecting **{HELP, Edit buffers}** also accesses the "point and shoot" style dialog box which lists each open edit buffer, 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.

Don't confuse edit buffer numbers with window ID numbers. Although they are often the same, they are independent. When you display a buffer in two or more windows, e.g. with **{VIEW, Toggle hex mode split}**, the buffer is attached to additional windows.

Use **{WINDOW, Switch}** to switch directly to a window.

See Also:

"Editing Multiple Files" in Chapter 4.
{FILE, Open}, **{FILE, Next buffer}**, **{FILE, Exit}**
{HELP, Edit buffers}
{WINDOW, Switch}

Next Buffer

Previous Buffer

Toggle round-robin style to the next or previous buffer (file).

Keystroke Equivalent:

<F6> is a keystroke macro for “Next buffer”.

<F5> is a keystroke macro for “Previous buffer”.

Full Description:

This is the easiest way to switch to another file when simultaneously editing multiple files. It toggles you, round-robin fashion, to the next/previous edit buffer (file). If necessary, select **{FILE, Next buffer}** (**<F6>** or toolbar) 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}**, but usually “Next buffer” or “Previous buffer” is simpler and quicker.

Notes:

These functions will also switch to empty edit buffers that are in use but have no file open in them. This can happen after you have used **{FILE, New}**, **{FILE, Buffer switch}** or when a command macro sets up additional edit buffers.

You can close buffers that are no longer needed, e.g. buffers that are empty or contain files you have finished editing, by switching to the buffer and selecting **{FILE, Close}** or **{WINDOW, Close}** (**<Ctrl-F4>**).

See Also:

“Editing Multiple Files” in Chapter 4.
{FILE, Open}, **{FILE, Buffer switch}**
{WINDOW, Next window}, {WINDOW, Previous window}

Print

Select the print dialog box to print the entire file or selected block.

Keystroke Equivalent:

<Ctrl-P>, This is a keystroke macro.

Full Description:

This item selects the Print dialog box. You can print the entire file, just the highlighted block or select the “PRINT.VDM macro”.

You can select whether the file/block is printed using the currently configured margins and print mode, or whether it is printed “raw” without margins.

You can also select the font and size used for printing. All characters are printed in the same font and size.

The **`print.vdm`** macro, as supplied, adds the filename, page number and date to the top of each printed page. It can optionally print rulers, line numbers and file positions (offsets).

Notes:

You can print multiple blocks of text on the same page by disabling the "**[ ] Auto-close (finish) print job**" option which leaves the print job open. You must then select **{FILE, Print}** again to print another block or finish and close the print job.

See Also:

"Printing" in Chapter 4.

Enable edit restore

Save the entire editing status when you exit.

Keystroke Equivalent: (None)

Full Description:

When enabled, VEDIT saves its entire status when you exit. Subsequently restarting VEDIT resumes your previous edit session, just as if you had never exited.

If you start VEDIT with files, the previous edit session status is not used. To invoke VEDIT without filenames and without restoring the previous edit session, use the "**-e**" option, e.g. "**vpw -e**".

See Also:

"Exiting VEDIT - Edit Session Restore" in Chapter 4.
{FILE, Exit}

Exit (save)

Save or abandon the file or files being edited and exit VEDIT.

Keystroke Equivalent:

<Alt-F4> or **<Alt-X>**, These are keystroke macros

Full Description:

"Exit" lets you selectively save or abandon each modified file, or save/abandon all files at once. It displays each altered buffer (file) and prompts whether it is to be saved or abandoned. It repeats this for each buffer and then exits VEDIT. For example, the prompt might be:

```
Save ERRATA.DOC ? [Yes] [No] [Save-all] [Quit-all]
```

[Yes] Saves the specified file (buffer); if the file has no assigned filename, you are prompted for one. If this is the last altered file, it exits VEDIT; otherwise it prompts for the next file.

[No] Abandons (quits) the specified file; the changes are not saved. If this is the last altered file, it exits VEDIT; otherwise it prompts for the next file.

[Save-all] Saves all files currently being edited and exits VEDIT without any further prompting. Note: it only saves those files (buffers) that have assigned filenames!

[Quit-all] Abandons all files currently being edited and exits VEDIT. If additional altered files are open, it prompts for verification.

[Cancel] Cancels the operation; same as pressing **<Esc>**.

Notes:

The **[Save-all]** and **[Quit-all]** selections may seem superfluous when only one file is being edited. However, they are necessary for a keystroke macro such as "[MENU] F X S" to work.

Selecting **[Save-all]** only saves text in those edit buffers that have assigned filenames. However, if the main edit buffer #1 has no filename assigned, it will prompt for the "Save as" filename.

Abandoning a file causes VEDIT to "forget" any changes you have made since the last time the file was saved. Use this capability with caution!

If **{FILE, Enable edit restore}** is enabled, VEDIT saves the entire edit session. Restarting VEDIT then resumes the editing.

{FILE, Exit} is identical to **{ESCAPE, Exit}**.

See Also:

"Exiting VEDIT" in Chapter 4.
{FILE, Save as}, **{FILE, Save}**

Recent File List

Recently edited files are listed and can be easily selected for further editing.

Keystroke Equivalent:

<Alt-F> followed by the number of the file.

Full Description:

The **{FILE}** menu in the Window version contains a MRU (Most Recently Used) list of recently edited files. Simply selecting a filename from the list re-opens the file.

Notes:

The number of recently edited files listed in the **{FILE}** can be changed by directly editing the **`vedit.ini`** file and changing the item "MaxMRU". The allowable values are "0" - "10".

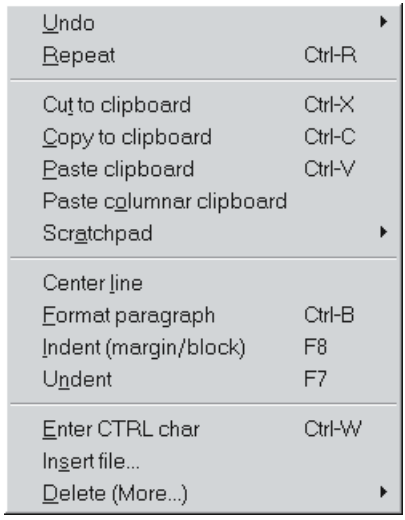
Edit Menu

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. It provides quick access to the Windows clipboard, the scratchpad, and lets you insert the contents of another file.



Repeat

Repeat next edit operation.

Keystroke Equivalent:

<Ctrl-R>, identical to **[REPEAT]**

Full Description:

"Repeat" repeats the following edit operation the specified number of times. Any edit operation, such as typing in a character, pressing an edit function key, selecting an item from a menu or even executing a keystroke macro, can be repeated.

➢ **To repeat an edit operation:**

1. Select **{EDIT, Repeat}** (**<Ctrl-R>**). You will see the following prompt:

   ```
   Enter repeat count:
   ```

2. Enter the count of how many times to repeat the upcoming operation. You can enter a numeric expression such as "(1250+540)/24".

3. At the prompt "Press key to repeat:" simply press the edit function or displayable character you want repeated. You can also repeatedly execute items from the main menu.

For example, to create the top of a box, you could press **[REPEAT]**, "**50**" <**Enter**> and "*****".

Notes:

The maximum repeat count defaults to 256 but can be changed with **{CONFIG, Misc, Maximum [REPEAT] count}**. This prevents you from accidentally repeating an operation so many times that it cannot be undone.

This menu item is usually only selected with a mouse; from the keyboard it is usually more convenient to simply press **[REPEAT]** (<**Ctrl-R**>).

When repeating menu items, ***YOU MUST*** select the items by typing the appropriate selection letters. Attempting to select items with the cursor keys will confuse the process and lead to undesirable 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-**\>) or <**Ctrl-Break**>.

See Also:

[REPEAT], **[REPEAT LAST]**

Cut to clipboard

Copy to clipboard

Copy or move the marked block of text to the Windows clipboard.

Keystroke Equivalent:

<**Ctrl-X**>, This is a keystroke macro.

<**Ctrl-C**>, This is a keystroke macro.

Full Description:

"Copy to clipboard" saves a copy of the marked (highlighted) text in the Windows clipboard, while "Cut to clipboard" also deletes the marked text from your file.

➢ **To save (cut or copy) text to the clipboard:**

1. Highlight the desired text. You can either set both markers or use the cursor as the second marker.
2. Select **{EDIT, Cut to clipboard}** (<**Ctrl-X**>) to cut (move) the block.

   Select **{EDIT, Copy to clipboard}** (<**Ctrl-C**>) to copy the block.

Notes:

The clipboard should only be used for exchanging data with other programs. The text registers (scratchpad) should be used for cut & paste operations within VEDIT. (The Clipboard does not handle columnar blocks as well as the text registers, and does not support binary data.)

The Windows "clipboard" is a temporary holding area for text to be exchanged with other programs. Text placed in the clipboard remains there until it is overwritten or you exit Windows. All Windows programs share the same clipboard. For example, you can copy a block of text from VEDIT to an E-mail program.

While most other editors require that you use the clipboard to cut and paste blocks within your file, we ***strongly suggest*** that you use VEDIT's text registers instead. The default text register 0, also called the ***scratchpad***, is just as easy to use. (The Clipboard does not handle columnar blocks as well as the text registers, and does not support binary data.)

Technical Notes:

VEDIT always copies data to the clipboard as "text" and can only paste from the clipboard when it contains "text", i.e. graphic bitmaps cannot be pasted into VEDIT. In "text" format, the clipboard cannot contain the "null" character (value 00). If you copy a block containing a "null" to the clipboard, anything past the "null" will appear to be truncated. Therefore, the clipboard cannot be used to exchange binary data between VEDIT and other programs.

DOS version: The Window clipboard is only available in Windows 95 and when running Windows 3.1 in enhanced mode.

See Also:

"Block Operations - The Windows Clipboard" in Chapter 4.
{EDIT, Scratchpad, Cut to scratchpad}, **{BLOCK, Copy to register}**.

Paste clipboard

Paste columnar clipboard

Insert the contents of the Windows clipboard as a "stream" or "columnar" block at the current cursor position.

Keystroke Equivalent:

<Ctrl-V>, This is a keystroke macro for **{EDIT, Paste clipboard}**.

Full Description:

"Paste Clipboard" inserts the contents of the Windows clipboard when it contains "text" type data; it does not insert graphical data.

You can paste the clipboard as either a "stream" or "columnar" block. Selecting "Paste clipboard" inserts the entire clipboard at the cursor position. Selecting "Paste columnar clipboard" inserts each line of the clipboard into successive lines of your file, each time starting at the current column.

Notes:

By default, the cursor will be positioned past the inserted text. Alternatively, set **{CONFIG, Emulation, Advance cursor past block insert}** to "No" to have the cursor remain at the beginning of the inserted text.

See Also:

"Block Operations - The Windows Clipboard" in Chapter 4.
{EDIT, Copy to clipboard}, **{BLOCK, Insert register}**

Center Line

Center the current line.

Keystroke Equivalent: (None)

Full Description:

"Center line" adds just enough space to the beginning of a line to center the line between column 1 and the right margin. If no explicit right margin is set, it defaults to the width of the current window.

➢ **To center a line of text:**

1. Make sure that the right margin is set correctly with **{CONFIG, Word processing, Right margin}**.
2. Select **{EDIT, Center line}**. This will center the current line. It also advances 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]** (**<Ctrl-R>**) and the desired count. Alternatively, follow this function with **[REPEAT LAST]** (**<Alt-R>**) to center additional lines.

See Also:

{CONFIG, Word processing, Right margin}

Format Paragraph

Format a paragraph between the left and right margins.

Keystroke Equivalent:

<Ctrl-B>, This is a keystroke macro.

Full Description:

This item (re)formats a paragraph to the current left and right margins. Optionally, the paragraph can also be "justified" with an even right edge.

➢ **To format (or re-format) a paragraph:**

1. Make sure that **{CONFIG, Word processing, Enable word wrap and formatting}** is enabled and that **{CONFIG, Word processing, Right margin}** is set correctly.
2. Also set the left margin if you want the paragraph indented. This margin can be set with **{CONFIG, Word processing, Left margin}**, but is typically set with **{EDIT, Indent}** (**<F8>**) and **{EDIT, Undent}** (**<F7>**).
3. Place the cursor anywhere in the paragraph and select **{EDIT, Format paragraph}** (**<Ctrl-B>**). After formatting, the cursor is advanced to the beginning of the next paragraph so that you can easily format several consecutive paragraphs.

Notes:

If **{CONFIG, Word processing, Enable word wrap and formatting}** is not enabled, you will be prompted for confirmation to enable it. This can save you the steps of going to the **{CONFIG}** menu.

Blank lines ***MUST*** be used to separate paragraphs from each other; just an indented line is not enough. Otherwise, formatting paragraphs will combine several paragraphs into one! The configuration parameter **Config_String(PARA_SEP)** (Chapter 8) determines which special characters at the beginning of a line also separate paragraphs from each other.

When formatting paragraphs, VEDIT normally trims trailing spaces, removes extra spaces from between words and leaves only a single space following ".", "!" and "?". If this is not desirable for your application, you can select other options with **{CONFIG, Word processing, Format paragraph options}**.

See Also:

"Word Processing - Formatting and Justifying Paragraphs" in Chapter 4.
{EDIT, Indent}, **{EDIT, Undent}**
{CONFIG, Word processing} sub-menu

Indent (margin/block)

Undent

Change the left margin or indent/undent the entire block.

Keystroke Equivalent:

<F8>, This is a keystroke macro.

<F7>, This is a keystroke macro.

Full Description:

If no block is highlighted, "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, the current line is indented to the new left margin.

If the cursor is within a highlighted block, it increases the indentation of all lines within the block by the amount of the *indent increment*. The left margin is not changed.

Similarly, "Undent" decreases the left margin or undents a highlighted block.

Notes:

The "indent increment" can be changed with **{CONFIG, Programming, Indent increment}**. The default value is "4".

By default, the indentation uses the optimal number of tabs and spaces. To only use spaces, set **{CONFIG, Tab/Fill, Expand <Tab> with spaces}** to "1", "3", "5" or "7".

See also:

"Indenting Text (Left Margin)" in Chapter 4
{CONFIG, Word processing, Left margin}, **{EDIT, Format paragraph}**

Enter CTRL Char

Enter the next key press literally into the text.

Keystroke Equivalent:

<Ctrl-Q>, This is identical to **[ENTER CTRL]**.

Full Description:

To enter a control character into the file, you must precede it with **[ENTER CTRL]**. Otherwise VEDIT treats a typed control character as an editing function or hot-key.

Certain control characters can be useful in your text. "Form-Feed" (**<Ctrl-L>**) and "Escape" (**<Esc>**) are frequently used in text files.

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>**). You are prompted for the control character on the status line.
4. Type the desired control character. For example, to insert **<Ctrl-L>**, hold down the **<Ctrl>** key while you type "**L**".

HINT: Alternatively, you can enter control and graphics characters anywhere (e.g. in the search dialog box) by holding down the **<Alt>** key and then typing its ASCII value on the numeric keypad. Then release the **<Alt>** key. This is a function of Windows/DOS and also works with most other programs.

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.

[ENTER CTRL] also works in the search and replace dialog boxes.

Characters can also be entered by ASCII value with **{MISC, ASCII table}**.

If you are entering or changing many control characters, it may be easier to switch to Hex-mode editing.

See Also:

"Screen Display and Keyboard Characters" in Chapter 4.
{MISC, ASCII table}, **{CONFIG, Characters/Cursors, Screen display mode}**
[ENTER CTRL] in Chapter 7.

Insert File

Insert an entire file at the cursor position.

Keystroke Equivalent: (None)

Full Description:

"Insert file" inserts an entire file at the current cursor position. This can be used to merge several files together or to start a newly created file with the 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 **{EDIT, Insert file}**.
3. You are prompted for the desired filename with the File dialog box.

Notes:

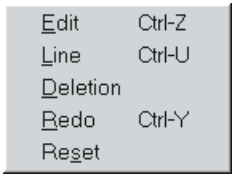
Assuming you have a file open, there is no limit to the size of the file that can be inserted. This is the reason you should use **{BLOCK, Write to disk}** and **{EDIT, Insert file}** when copying or moving very large blocks of text between files. If necessary, use **{FILE, Save as}** to open the destination file before inserting a large file.

After inserting a file, the cursor will normally be positioned past the inserted text. However, 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.

Undo (Sub-menu)

Introduction:

VEDIT can undo most editing operations including cursor movements, insertions, deletions and search/replace. You can undo operations keystroke-by-keystroke, line-by-line or deletion-by-deletion. The "redo" capability lets you "undo" the undo.



See Also:

The topic "Undo and Redo" in Chapter 4.

Edit

Undo a single keystroke or editing operation.

Keystroke Equivalent:

<Ctrl-Z> or **<Alt-Bksp>**, These are keystroke macros.

Full Description:

Each time "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 opened the file, VEDIT will know that the file has not been altered.

Line

Undo all changes made to the current line.

Keystroke Equivalent:

<Ctrl-Shift-Z>, This is a keystroke macro.

Full Description:

All changes made to the current line of text can be reversed in one step with this item. Immediately selecting this item 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 this item again to also undo this line.

Deletion

Insert the last text deletion(s).

Keystroke Equivalent: (None)

Full Description:

Up to five blocks of text deleted with **[ERASE EOL]**, **[ERASE LINE]** and **{BLOCK, Edit/translate, Block delete}** can be re-inserted at any place in your file using this function. It 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, **{EDIT, 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").

Notes:

Technically, **{EDIT, Undo, Deletion}** is not an undo function; it is an editing function that inserts a previously deleted block of text anywhere in the file. Therefore, you can undo **{EDIT, Undo, Deletion}** with **{EDIT, Undo, Edit}**.

When enough memory is available, VEDIT can undo up to 256K of deleted text. Only the last five deleted lines and blocks can be re-inserted with **{EDIT, Undo, Deletion}**. If the last five deletions exceed 256K, it may not be possible to re-insert more than one or two deleted blocks.

A block deletion giving the confirmation prompt "Cannot undo this operation! Proceed anyway?" cannot be re-inserted or otherwise undone.

Redo

Redo the effects of the last undo operation.

Keystroke Equivalent:

<Ctrl-Y>, These is a keystroke macros.

Full Description:

Each "Redo" reverses the effects of the previous "undo" operation. This is useful when you have gone too far with "undo" and want to restore your editing changes.

Redo is only available while performing Undo operations. Once you stop using Undo and perform normal editing, redo is no longer available. After switching buffers (files) you can no longer redo changes to previous buffers. You can Redo as many operations as you can Undo.

Each redo only restores one editing operation at a time — each redo reverses the effects of one **{EDIT, Undo, Edit}**. Since **{EDIT, Undo, Line}** can undo several (many) edit operations at a time, it may take several (or many) **{EDIT, Undo, Redo}** to reverse the effects of one **{EDIT, Undo, Line}**.

Reset

Reset the undo memory.

Keystroke Equivalent: (None)

Full Description:

"Reset" clears the undo memory (and resets all undo stacks). If you now make additional edit changes, you will not be able to undo any further back than when you selected this item. You can now try out different editing scenarios and if they do not work, use **[REPEAT] 255 {EDIT, Undo, Line}** to quickly undo back to this point.

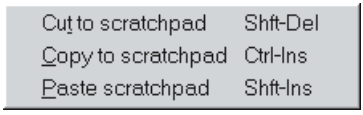
Scratchpad (Sub-menu)

Introduction:

The *scratchpad* is a temporary holding area for cut and paste operations within VEDIT. It is also referred to as the default text register "0"; VEDIT has over 100 text registers. Text you save stays in the scratchpad until you either save a different block or exit VEDIT.

The scratchpad is ***not*** the same as the Windows clipboard; but you can think of it as VEDIT's internal clipboard. The scratchpad's advantages include being able to handle columnar blocks better and support for binary files.

- Use the clipboard for exchanging text with other Windows programs.
- Use the scratchpad for cut and paste operations within VEDIT.



Cut to scratchpad

Copy to scratchpad

Copy or cut (move) a single line or the highlighted block of text to the "scratchpad" — text register "0".

Keystroke Equivalent:

<**Shft-Del**>, This is a keystroke macro.

<**Ctrl-Ins**>, This is a keystroke macro.

Alternatively, double-press the keys assigned to **{BLOCK, Move to register}** (default: <**Numpad-**>) and **{BLOCK, Copy to register**} (<**Numpad+**>) to cut/copy to the scratchpad.

Full Description:

"Copy to scratchpad" saves a copy of the highlighted text in the scratchpad, while "Cut to scratchpad" also deletes the highlighted text from the file.

If no block is highlighted, these functions copy/cut the current line to the scratchpad. The line is stored as a "line block"; when inserted it is always inserted at the beginning of the current line.

These functions are used in the same way as **{EDIT, Cut/copy to clipboard}**.

Notes:

For maximum flexibility, experienced users may want to skip the scratchpad functions in the **{EDIT}** menu and use the text register functions in the

{BLOCK} menu instead. With these functions you can optionally append to a text register and fill the original block with spaces.

The default hot-keys for the scratchpad functions are the "old style" clipboard keys. (Everyone found them difficult to remember.)

HINT: We ***highly*** suggest accessing the scratchpad by double-pressing the hot keys assigned to the Copy/Move/Insert text register functions in the **{BLOCK}** menu, e.g. the **<Numpad>** keys. Double-pressing the hot-key for any text register prompt selects the default scratchpad.

The toolbar also has buttons for selecting the scratchpad.

See the "Notes" for **{BLOCK, Copy to register}**.

See Also:

"Block Operations - Scratchpad and Text Registers" in Chapter 4.
{BLOCK, Copy to register}

Paste scratchpad

Insert the contents of the scratchpad at the current cursor position.

Keystroke Equivalent:

<**Shft-Ins**>, This is a keystroke macro.

Alternatively, double-press the key assigned to **{BLOCK, Insert register}** (default: <**Numpad***>) to insert the scratchpad.

Full Description:

This item inserts the contents of the scratchpad - text register "0" - at the current cursor location. If the scratchpad is empty, this command has no effect.

Notes:

The scratchpad will be inserted as the same type of block as it was saved. A line block will be inserted before the current line. A columnar block will be inserted on the current line and at the starting column on successive lines.

This item normally inserts text regardless of the "Insert mode" status for normal editing. **{BLOCK, Insert register}** has an option for overwriting the existing text. However, in "overstrike-only" mode, e.g. when editing binary/data files, the existing text at the cursor will be overwritten.

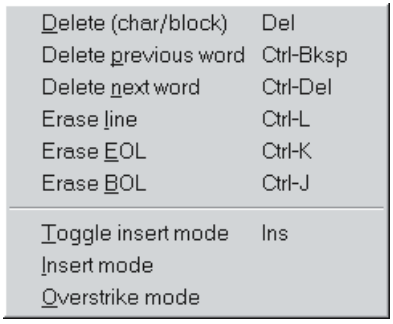
By default, the cursor will be positioned past the inserted text. Alternatively, set **{CONFIG, Emulation, Advance cursor past block insert}** to "No" to have the cursor remain at the beginning of the inserted text.

See Also:

"Block Operations - The Scratchpad and Text Registers" in Chapter 4.
{BLOCK, Insert registers}

Delete (Sub-menu)

This sub-menu contains items for deleting text and changing the state of Insert mode.



Delete (char/block)

Delete the current character or highlighted block of text.

Keystroke Equivalent:

<**Del**>, identical to **[DELETE]**

Full Description:

Deletes the character at the cursor. At the end of a line, it deletes the "newline". (For DOS/Windows text files, the "newline" consists of "Carriage-return" and "Line-feed".)

If a block is currently highlighted and the cursor is within the block (or immediately past it), it deletes the block; same as **{BLOCK, Edit/Translate, Block Delete}**.

Notes:

{CONFIG, Emulation, Special emulation modes} controls whether this command deletes "newlines" and highlighted blocks.

{BLOCK, Edit/Translate, Block Delete} always deletes the highlighted block, regardless of how this configuration parameter is set.

This menu item is usually only selected with a mouse; from the keyboard it is usually more convenient to simply press **[DELETE]** (<**Del**>).

See Also:

"Block Operations - Marking a Block of Text" in Chapter 4.
{BLOCK, Edit/Translate, Block Delete}
[DELETE] in Chapter 7.

Delete Previous Word

Delete Next Word

Delete the previous/next word and the space between words.

Keystroke Equivalent:

<**Ctrl-Del**>, identical to **[DEL NEXT WORD]**

<**Ctrl-Bksp**>, identical to **[DEL PREV WORD]**

Full Description:

"Delete previous word" deletes the word, or portion of a word, or whitespace to the left of the cursor. Pressing it again deletes the next whitespace or word.

Similarly, "Delete next word" deletes the word, portion of a word, or whitespace to the right of the cursor.

Notes:

The configuration parameter **Config_String(WORD_SEP)** (Chapter 8) specifies which characters, in addition to spaces, tabs, newlines and control characters, separate words from each other.

See Also:

The topic "Word Processing" in Chapter 4.
[DEL NEXT WORD] in Chapter 7.

Erase Line

Erase EOL

Erase BOL

Erase (delete) a partial or entire line.

Keystroke Equivalent:

<**Ctrl-L**>, identical to **[ERASE LINE]**

<**Ctrl-K**>, identical to **[ERASE EOL]**

<**Ctrl-J**>, identical to **[ERASE BOL]**

Full Description:

"Erase Line" deletes the entire line of text on which the cursor is located. It also deletes the "newline" character(s) at the end of the line and closes up the line on the screen.

"Erase EOL" deletes all characters from the cursor position to the end of the line. It does not delete the "newline" character(s) at the end of the line.

"Erase BOL" deletes all characters from the beginning of the line up to, but not including, the cursor position.

Notes:

These menu items are usually only selected with a mouse; from the keyboard it is usually more convenient to simply press the corresponding key.

{EDIT, Undo, Deletion} re-inserts deleted lines in a different location. Up to five lines can be re-inserted, in reverse order, in this way.

HINTS: If you set **{CONFIG, Emulation, Alt/Ctrl/Shift key shortcuts}** to "15", you can also delete lines by pressing **<Ctrl-Shift>**.

You can also delete a line by ***moving*** it to the scratchpad with **{BLOCK, Move to register}**; this assumes no block is highlighted. Since the "Normal" hot-key is the easy-to-reach **<Numpad->** on the numeric keypad, you can double-press this key to erase lines when you don't have anything saved in the scratchpad.

See Also:

[ERASE LINE], **[ERASE BOL]** and **[ERASE EOL]** in Chapter 7.

Toggle Insert Mode

Insert Mode

Overstrike Mode

Toggle or set Insert/Overstrike mode.

Keystroke Equivalent:

<**Ins**>, identical to **[INSERT TOGGLE]**

Full Description:

<**Ins**> is usually used to toggle between "Insert" and "Overstrike" modes.

"Insert mode" and "Overstrike mode" explicitly put VEDIT into Insert or Overstrike mode. The primary purpose for these items is for keystroke macros that require a particular mode.

Notes:

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.

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, Delete, 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, Delete, Insert mode}
[INSERT TOGGLE]

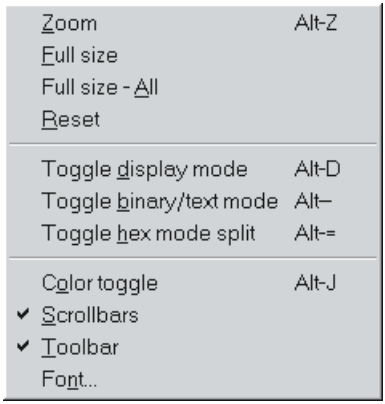
View Menu

Keystroke Equivalent:

<**Alt-V**>, This is a keystroke macro.

Introduction:

The **{VIEW}** menu lets you change the window view by zooming/dezooming a window, resetting all windows, and changing the window text font. You can also toggle through a wide range of display, binary and hex editing modes.



Zoom

Zoom / de-zoom the current window.

Keystroke Equivalent:

<**Alt-Z**>, This is a keystroke macro.

Full Description:

"Zoom" expands the current window to its maximum possible size, thereby displaying more of the file. Each window that you subsequently switch to will also be zoomed.

Selecting "Zoom" again restores the window to its normal size.

Notes:

Clicking the mouse on the window's "zoom" button is the same as selecting this item.

The VEDIT program also has an "overall zoom button" which expands it to the full screen. Click both VEDIT's zoom button and the window's zoom button to expand a window to full screen size.

Some VEDIT macros create special “reserved” windows at the top or bottom of the screen that always remain visible. They effectively reduce the maximum size of zoomed windows, just like the status line, toolbar and menu.

See Also:

“Windows - Zooming a Window” in Chapter 4.
{WINDOW, Switch}

Full Size

Full Size - All

Expand the size of the current window or all windows to full-size.

Keystroke Equivalent: (None)

Full Description:

“Full size” is similar to manually stretching a window’s borders to the maximum possible size. However, a full-size window will change size if you change VEDIT’s overall window size. A window remains full-size until you explicitly resize, tile or cascade it.

This lets you edit in the largest possible window without having to zoom all windows.

“Full size - all” makes all editing windows full-size.

Notes:

“Full size” is not the same as “zoomed”. When “zoomed”, each window you switch to will be zoomed, until you “dezoom”. In contrast, some of your editing windows can be full-size, while other are not; they can be custom sized or tiled.

{CONFIG, Display options, Auto-create window style} determines if newly created window are initially full-size or cascaded.

Some VEDIT macros create “reserved” windows at the top or bottom of the screen that always remain visible. They effectively reduce the size of full-size windows.

See Also:

“Windows - Full-Size Windows” in Chapter 4.

Reset

Reset all windows to full-size or cascaded; remove extra windows.

Keystroke Equivalent: (None)

Full Description:

“Reset” initializes the windows, resets the display mode and window colors (attributes). The buffers (files) are not affected. No text is lost!

Following "Reset" each open buffer (file) will have one corresponding window; it will be either full-size or cascaded, depending upon the setting of **{CONFIG, Display options, Auto-create window style}**. All other windows will be removed.

DOS, UNIX and QNX Versions:

Assuming **{CONFIG, Display options, Auto-create window style}** is enabled (default), each open buffer (file) will have one corresponding window of full screen size; all other windows will have been deleted. If disabled, only the main window "1" will exist; all buffers will then share the same window.

Notes:

This function is useful for getting VEDIT back to a reasonable state after you have split many windows, changed display modes and switched window colors.

Toggle Display Mode

Toggle the current window through ASCII, Hex and EBCDIC display modes.

Keystroke Equivalent:

<Alt-D>, This is a keystroke macro.

Full Description:

This item toggles the current window through five ASCII display modes, hexadecimal, octal and EBCDIC. The ASCII display modes let you view control and graphics characters in different ways.

It is easiest to understand these modes by toggling through all modes while viewing a file containing some control and graphics characters.

➢ **To toggle the current window through all display modes:**

1. Select **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar). This will toggle to the next mode.
2. With the normal keyboard layout, repeatedly press (**<Alt-D>**) until you have toggled through all modes. Alternatively, if the **<Alt-D>** hot-key is not defined, you can repeatedly press **[REPEAT LAST]** (**<Alt-R>**).

Notes:

If you toggle an ASCII file into EBCDIC mode, it will display as gibberish. Conversely, an EBCDIC file displays as gibberish in ASCII mode. For this reason the status line displays "EBCDIC" when in this mode.

The status line temporarily displays a message indicating which mode you are in.

You can restore the screen to its normal mode by selecting**{VIEW, Reset}**. This will also remove extra windows.

You can select a particular display mode with **{CONFIG, Characters/Cursors, Screen display mode}**. This will affect the current window and all newly created windows.

See Also:

"Screen Display & Keyboard Characters - Display Modes" in Chapter 4.
{VIEW, Toggle Hex-mode split}
{CONFIG, Characters/Cursors, Screen display mode}.

Toggle Binary/Text Mode

Toggle the current file between 16 and 64 bytes per line, and the original file type.

Keystroke Equivalent:

<Alt—>, This is a keystroke macro.

Full Description:

Toggles the current file's "file type" to binary (i.e. fixed-length records) with 16 or 64 bytes per line and back to the original file type. This is useful for editing binary, data and other non-standard files, especially in hexadecimal.

Notes:

This function does not alter the file. Use the **{BLOCK, Convert newlines}** sub-menu to translate between different file types or from fixed-length records to newlines.

The status line temporarily displays a message indicating which mode you are in.

See Also:

"File Types - Binary/Data Files" in Chapter 4.

Toggle Hex Mode Split

Split the current window into ASCII and Hex-mode windows.

Keystroke Equivalent:

<Alt-=>, This is a keystroke macro.

Full Description:

This item splits the current window into two vertical windows. The left window displays the file in hexadecimal and the right window in ASCII. You can edit in either window; both windows will update together.

Selected again, it changes the file type to Binary-16 for easier hex mode editing. Selected again, it toggles back to the original editing mode.

The ASCII window uses display mode "4" in which all characters, including the "newline" <LF>, <CR> and <Tab> character are displayed literally (using the IBM PC character set). The hex window uses display mode "8" to display all characters in hexadecimal.

Use **{WINDOW, Switch}** (**<Alt-F5>**), **{WINDOW, Next window}** (**<Ctrl-F6>**) or the mouse to switch between the windows. In the hex-mode window, new characters must be entered in hexadecimal.

Notes:

The ASCII and the hex-mode windows will scroll together as you move about in the file.

You can change the display mode used in either window by switching to that window and using **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar). Therefore, you can easily edit in Hex and EBCDIC.

This function is implemented by the **`hexsplit.vdm`** macro.

See Also:

"Hexadecimal and Octal Mode Editing" in Chapter 4.

VGA 25/28/50 Line Toggle (DOS Only)

Switch between 25, 28 and 50 line VGA modes. (Optionally 30, 34 and 60 line modes.)

Keystroke Equivalent:

<Alt-T>, This is a keystroke macro.

Full Description:

"VGA 25/28/50 line toggle" switches most VGA compatible display adapters between 25, 28 and 50 lines of text. If **Config(H_VGA_TYPE)** is set to "7", it also toggles to 30, 34 and 60 line modes.

Upon startup, VEDIT normally adjusts itself to the current size of the display.

If your display adapter and monitor (possibly with additional software) are capable of displaying more lines or columns of text, VEDIT will take advantage of it.

Notes:

Some systems only support 25 and 50 line modes and may hang when attempting to switch to 28 line mode. The default VGA text mode uses 400 scan lines; by switching to 480 scan lines, 30, 34 and 60 lines can be displayed. The allowable modes are selected with **{CONFIG, Misc, Enable VGA 28/30/34/50/60}**.

Appendix C (Application Notes) lists a keystroke macro for switching into 132 column mode. This item can only toggle between the standard 80 characters/line modes.

To force VEDIT to start in 50 line mode, set **Config(S_N_LINE)** in the **`vedit.cfg`** file to "50".

Color Toggle

Select the next suggested color combination.

Keystroke Equivalent:

<Alt-J>, This is a keystroke macro.

Full Description:

This function toggles the current window's text color through about twenty combinations. This provides some variety as you work and can be used to make some windows visually stand out.

The initial text color is set according to **{CONFIG, Editing colors, Edited text}**. If only one window is open, this item also changes the configuration value; however it does not automatically save it.

➢ **To change the window's text color and save it as the configured value:**

1. Ensure that only one window is open. This method only works when only one window is open.
2. Repeatedly select **{MISC, Color toggle}** (**<Alt-J>**) until the text appears in the desired color.
3. Select **{CONFIG, Save config}** if you want to make the new color combination permanent.

Notes:

Additional color combinations can be selected with **{CONFIG, Editing colors, Edited text}**. (Non-Windows versions: **{CONFIG, Colors, Windows and editing, Edited text}**).

The overall color scheme, including coordinated syntax highlighting colors, can be selected by running the `color.vdm` macro via **{MISC, More macros, Color}** or by selecting "Color scheme" in the default **{USER}** menu.

Scroll bars (Windows only)

Toggle the scroll bars in all windows on and off.

Keystroke Equivalent: (None)

Full Description:

Toggles the scroll bars in all windows on and off. Scroll bars are usually desirable, but if you must see as many columns and lines as possible, you can turn them off.

To configure whether the scroll bars are enabled when VEDIT starts up, set **{CONFIG, Display options, Enable scroll bars}** and then select **{CONFIG, Save config}** to save the new setting.

Notes:

The vertical scroll bar is only displayed when the file (buffer) contains enough lines to require scrolling.

Toolbar (Windows only)

Toggle the VEDIT toolbar on and off.

Keystroke Equivalent: (None)

Full Description:

The Windows version of VEDIT normally displays a toolbar with icons. The toolbar is an easy way to access commonly used menu items with the mouse. If desired, the toolbar can be turned off to make more screen area available for editing.

To configure whether the toolbar is enabled when VEDIT starts up, set **{CONFIG, Display options, Enable toolbar}** and then select **{CONFIG, Save config}** to save the new setting.

Notes:

Clicking the right mouse button also pops up a menu of commonly used items.

Font (Windows only)

Change the display font.

Keystroke Equivalent: (None)

Full Description:

"Font" changes the display font and size used for all editing windows. It displays the standard font selection dialog box; however, only "fixed width" fonts are listed and supported by VEDIT.

The fonts "VEDIT Oem", "VEDIT Ansi", "Fixedsys" and "Terminal" look and work best with VEDIT; each comes in several sizes. The True-Type font "Courier New" can be set to any desired size, but is of lower quality and slows down screen updates.

The fonts "VEDIT Oem" and "Terminal" use the OEM (IBM PC) character set. "VEDIT Ansi", "Fixedsys", "Courier" and most other fonts use the ANSI character set.

The custom "VEDIT" fonts display control characters in a useful mnemonic way, e.g. Ctrl-A is displayed as a small single-character "^**A**". Most other ANSI fonts display control characters as a useless solid block. Other OEM fonts display control characters as on the original IBM PC, e.g. Ctrl-A is displayed as the "smiley face".

The last selected font is saved into the **`vedit.ini`** file and the same font will be used the next time VEDIT is run.

See Also:

"User Interface - Selecting Display Fonts" in Chapter 4.

Block Menu

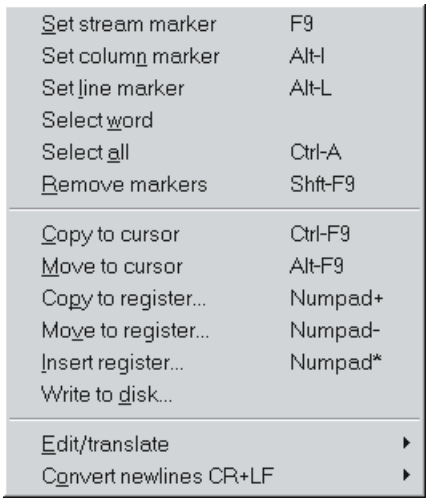
Keystroke Equivalent:

<Alt-B>, 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.



Set Stream Marker

Mark the beginning and end of a “stream” block of text.

Keystroke Equivalents:

<F9>, this is a keystroke macro. Alternatively, **{BLOCK, Copy to cursor}** can be used to set block markers.

Full 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.

Small block are often simply marked by dragging the mouse over them or by holding down the **<Shift>** key while moving the cursor. However, to mark a larger block, it is often best to explicitly mark the beginning and end of the block.

➢ **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 stream marker}** (**<F9>** or toolbar). Note the message "1-END" on the status line.

   Alternatively, select a columnar block with **{BLOCK, Set column marker}** (**<Alt-I>**) or a line block with **{BLOCK, Set line marker}** (**<Alt-L>**).
3. As you now move the cursor to the end of the block, intervening text will be highlighted to define the block.
4. Set the second marker by again selecting **{BLOCK, Set stream marker}** (**<F9>**). Note the message "BLOCK" 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.

After marking a block, you can change its size and/or the type of block. Move the cursor to the desired end of the block and then select "Set stream marker", "Set column marker" or "Set line marker" from the **{BLOCK}** menu.

➢ **To remove (clear) the block markers at any time:**

1. Press **[CANCEL]** **<Ctrl-****>**) or **<Ctrl-Break>**.

-OR-

Select **{BLOCK, Remove markers}** (**<Shift-F9>** or toolbar).

-OR-

Select **{ESCAPE, Remove block markers}**.

-OR-

Simultaneously press both mouse buttons.

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. See "Block Operations" in Chapter 4 (Editing Guide) for a more complete explanation.

Search/replace operations can be restricted to the highlighted block. See "Searching within a Block" described under **{SEARCH, Search}** in this chapter.

The block markers can also be set or removed with the mouse. The block markers can also be set by holding down the **<Shift>** key and moving the cursor.

See Also:

"Mouse Support - Block Operations" in Chapter 4.
"Block Operations - Marking a Block of Text" in Chapter 4.
{BLOCK, Copy to cursor}, **{BLOCK, Set column marker}**

Set Column Marker

Mark the beginning and end of a "columnar" block of text.

Keystroke, Edit Function Equivalent:

<Alt-I>, this is a keystroke macro.

Full Description:

"Set column marker" is similar to "Set stream 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 stream marker}** will change the block from a "stream" 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 positioning 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 columnar block and will insert it as a columnar block.

The topic "Block Operations - Columnar Blocks" in Chapter 4 (Editing Guide) describes columnar blocks in detail. It describes how VEDIT pads and trims spaces at the ends of lines and deals with Tab characters.

To mark a columnar block with the mouse, hold down the **<Alt>** key while dragging the mouse over the desired text.

See Also:

"Block Operations - Marking a Block of Text" in Chapter 4.
{BLOCK, Set stream marker}

Set Line Marker

Mark the beginning and end of a "line" block of text.

Keystroke, Edit Function Equivalent:

<Alt-L>, this is a keystroke macro.

Full Description:

"Set line marker" is similar to "Set stream marker" except that it selects entire lines in the file including the "newline" at the end of each text line.

Notes:

After copying a line block to a text register, VEDIT remembers that the register contains a line block and will insert it at the beginning of the current line instead of exactly at the cursor position.

See Also:

"Block Operations - Marking a Block of Text" in Chapter 4.
{BLOCK, Set stream marker}

Select Word

Select the current word as a block.

Keystroke Equivalent: (None)

Full Description:

This item selects and highlights the current word as a block. If the cursor is between words, the previous word and all whitespace up to the cursor is highlighted.

Notes:

It is usually easier to select a word by double clicking on it with the mouse.

The configuration parameter **Config_String(WORD_SEP)** (Chapter 8) determines which characters separate words from each other.

See Also:

"Mouse Support - Block Operations" in Chapter 4.
"Word Processing - Definition of Word and Paragraph" in Chapter 4.
{BLOCK, Set stream marker}

Select All

Select the entire file as a stream block.

Keystroke Equivalent: (None)

Full Description:

This item marks the entire file as a block of text. It is equivalent to using "Set stream marker" at both ends of the file. It is primarily used when the file is about to be processed with the **{BLOCK, Edit/translate}** sub-menu.

See Also:

"Block Operations - Marking a Block of Text" in Chapter 4.
{BLOCK, Set stream marker}

Remove Markers

Remove any existing block markers.

Keystroke Equivalent:

<**Shft-F9**>, this is a keystroke macro. However, **[CANCEL]** (<**Ctrl-**\>) or <**Ctrl-Break**> are usually used to remove markers.

This function is also on the toolbar.

Full 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.

Notes:

It is usually easier to remove the block markers by pressing **[CANCEL]** (<**Ctrl-**\>) or (<**Ctrl-Break**>) or simultaneously pressing both mouse buttons.

You can also perform **[CANCEL]** by just double-pressing <**Ctrl**>; this assumes **{CONFIG, Emulation, Alt/Ctrl/Shift key shortcut modes}** is enabled.

This item is also available in the **{ESCAPE}** menu and on the toolbar.

See Also:

"Block Operations - VEDIT's Blocks are Persistent" in Chapter 4.
{BLOCK, Set stream marker}
[CANCEL]

Copy to Cursor

Move to Cursor

Copy or move the highlighted block of text to the current cursor position, without using the scratchpad or clipboard.

Keystroke Equivalent:

<**Ctrl-F9**>, This is a keystroke macro.

<**Alt-F9**>, This is a keystroke macro.

Full Description:

"Copy to cursor" copies a highlighted block of text from one location in your file to another. It does not delete or otherwise affect the original block. It requires fewer steps than using the scratchpad (or clipboard) to copy and paste a block.

"Move to cursor" is similar except that the original block 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.

If a block is not yet highlighted, these functions set the first and second block markers, same as **{BLOCK, Set stream marker}**. Only when both block markers are set, do they copy/move the block. This lets them be used as "smart" keys that both mark the block and copy/move it.

➢ **To directly copy/move a block of text:**

1. Highlight the desired text, setting both block markers. If desired, the hot-keys for these functions can be used to set the block markers.
2. Move the cursor to the location in the file to which you would like the text copied. The text will be copied in front of the cursor.
3. Select **{BLOCK, Copy to cursor}** (**<Ctrl-F9>**) to copy the block.

   Alternatively, select **{BLOCK, Move to cursor}** (**<Alt-F9>**) to move the block.

Notes:

{BLOCK, Copy to cursor} performs a convenient combination of setting the block marker and copying a block 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 to cursor}** can also be used to mark a block of text for other operations.

These functions normally clear the block markers. However, by setting the configuration parameter **{CONFIG, Emulation, Block marker emulation mode}** to "1" or "2", the block will be highlighted in its new location.

In "overstrike-only" mode, e.g. when editing binary/data files, "Move to cursor" causes the original text to be filled (overwritten) with the configurable "block fill" character, typically spaces. The existing text at the cursor is also overwritten.

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 (scratchpad) and then insert the register where needed.

You can also directly copy or move blocks using the mouse.

See Also:

"Mouse Support - Block Operations" in Chapter 4.
"Block Operations - Direct Block Copy/Move" in Chapter 4.
{BLOCK, Set stream marker} {BLOCK, Copy to register}

Copy to Register

Move to Register

Save (copy or move) the current line or highlighted block to a text register.

Keystroke Equivalent:

<Numpad+> or **<Ctrl-F11>**, identical to **[T-REG COPY]**.

<Numpad-> or **<Alt-F11>**, identical to **[T-REG MOVE]**.

Full Description:

A *text register* is a temporary holding area for blocks of text. 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.

"Copy to register" saves a copy of the highlighted text in a register while "Move to register" also deletes the highlighted text from your file as it is placed in the text register.

If no block is highlighted, these functions copy/move the current line to the selected register.

➢ **To save (copy or move) 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}** (**<Numpad+>**) to copy the block.

   Select **{BLOCK, Move to register}** (**<Numpad->**) to move the block.

   You are prompted for the register number:

   ```
   Register number:
   ```
3. To select the default text register "0" (the scratchpad), just press **<Enter>** or press the hot-key again.

   Otherwise, enter the desired text register number "0" through "100".

HINT: When prompted for a text register number, press any function/control key to select the default "scratchpad" register "0". For example, simply double-press the hot-key for **[T-REG COPY]** to copy the highlighted block to register "0".

To append the block to the existing contents of a text register or insert the block at the beginning of the existing contents, select the **( ) Append** or **( ) Insert** option in the dialog box.

Notes:

The **<Numpad>** hot-keys are usually more convenient, but laptop users can use **<F11>** instead.

In "overstrike-only" mode, e.g. when editing binary/data files, "Move to register" causes the original text to be filled (overwritten) with the configurable "block fill" character, typically spaces.

Text register contents are lost when you exit VEDIT. However, if you restore a previous edit session, the text register contents are also restored.

There is a limit to the amount of text that can be saved in the text registers. This is usually between 40,000 and 60,000 characters, but can be much less if VEDIT has limited memory available to it. If possible, try working with several smaller pieces of text.

Huge blocks can be copied by first writing them to disk with **{BLOCK, Write to disk}**. Then position the cursor at the destination, perhaps after opening another file, and insert the block with **{EDIT, Insert file}**.

DOS, UNIX and QNX Versions:

The text register dialog box has a "terse" selection option. To select from the first ten registers, simply press "0" through "9". To select other registers, first type "**.**" (period), type the number and then press **<Enter>**. For example, to select register 10, enter "**.10**".

Note the help line at the bottom of the screen; use the on-line help for more details. This is controlled with **{CONFIG, Misc, Full/Terse dialog box options}**.

See Also:

"Block Operations - Scratchpad and Text Registers" in Chapter 4.
{EDIT, Scratchpad, Cut to scratchpad}
{BLOCK, Set stream marker}, **{BLOCK, Copy to cursor}**

Insert Register

Insert the contents of a text register at the current cursor position.

Keystroke Equivalent:

<Numpad*> or **<F11>**, identical to **[T-REG INSERT]**.

Full Description:

"Insert register" inserts the contents of a text register at the current cursor location. If the text register is empty, this command has no effect.

➢ **To insert the contents of a text register:**

1. Position the cursor at the desired location in the file. The register contents will be inserted just before the cursor.
2. Select **{BLOCK, Insert register}** (**<Numpad*>** or **<F11>**). You are prompted for the register number:

   ```
   Register number:
   ```

3. To select the default text register 0 (the scratchpad), just press **<Enter>** or press the hot-key again.

   Otherwise, enter the desired text register number "0" through "100".

You can optionally have the register overwrite the existing text. Select the **"[ ] Overwrite**" option before entering the register number.

Notes:

The register will be inserted as the same type of block as it was saved. A line block will be inserted before the current line. A columnar block will be inserted on the current line and at the starting column on successive lines.

In "overstrike-only" mode, e.g. when editing binary/data files, the overwrite option is automatically selected.

You can insert a text register even when a block is currently highlighted.

By default, the cursor will be positioned past the inserted text. Alternatively, set **{CONFIG, Emulation, Advance cursor past block insert}** to "No" to have the cursor remain at the beginning of the inserted text.

You can also insert the contents of an edit buffer. At the prompt enter the buffer number followed by "**+BUFFER**". For example, to insert buffer #5, enter "**.5+buffer**".

See the Hints and Notes for {BLOCK, Copy to register}.

See Also:

"Block Operations - Scratchpad and Text Registers" in Chapter 4.
{EDIT, Scratchpad, Paste scratchpad}
{BLOCK, Copy to register}, **{BLOCK, Move to register}**

Write to Disk

Write the highlighted block of text to a disk file.

Keystroke Equivalent: (None)

Full Description:

A text block of any size may be saved to disk. You can use this function to split a large file into smaller files, or to save a text block to disk for future use.

Unlike saving text in a text register, text written to disk will be available after you exit 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 "stream", "columnar" or "line" block.
2. Select **{BLOCK, Write to disk}**. You are prompted for the desired filename. Either enter a new filename or select an existing file to be overwritten.

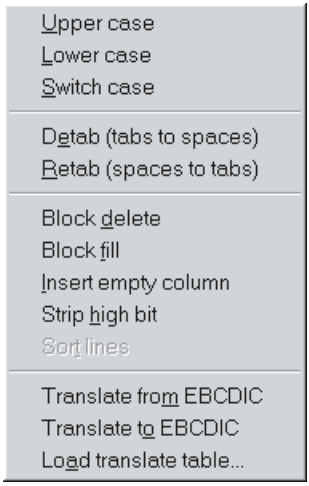
See Also:

"Block Operations - Cut & Paste Huge Blocks" in Chapter 4.
{EDIT, Insert file}

Edit/Translate (Sub-menu)

Introduction:

This menu processes the currently marked block: change case, fill with spaces, detab/retab, or translate to/from another character set.



Upper Case

Lower Case

Switch Case

Change the case of the current character (of limited use) or all characters in a highlighted block (very useful).

Keystroke Equivalent: (None)

Full Description:

If the cursor is in a highlighted block of text, the case of all letters in the block is changed. Otherwise, these functions simply change the case of the current letter and advance the cursor to the next character. Non-letters are not affected.

After changing the case of a single character, you can repeatedly press (or hold down) **[REPEAT LAST]** (**<Alt-R>**) to change additional characters.

These functions work with a stream, columnar or line block of any size.

See Also:

{CONFIG, Programming, Lower/Upper case key conversion}

The file **`keymac.lib`** contains a keystroke macro for switching the case of all letters from the cursor to the end of the line.

Detab (Tabs to spaces)

Retab (Spaces to tabs)

Convert Tab characters in a block to spaces; convert spaces to the optimum number of tabs.

Keystroke Equivalent: (None)

Full Description:

If a block is marked, "Detab" converts all tab characters in the marked block to spaces according to the currently set tab stops.

If no block is marked, the entire file is detabbed. However, any tab characters within single or double quotes are not converted. This is useful when editing C and other programs; literal tab characters may occur within quoted strings and should not be converted to spaces.

Similarly, if a block is marked, "Retab" converts sequences of space characters in the marked block to the optimal number of tabs and spaces according to the currently set tab stops.

If no block is marked, the entire file is retabbed. However, any spaces within single or double quotes are not converted. This is better for C and other programs.

➢ **To convert all Tabs in a file to spaces:**

1. If necessary, set the correct tab stops with **{CONFIG, Tab stops}**. The default for Windows/DOS is every 8 columns.
2. Select **{BLOCK, Select all}** to mark the entire file as a block.
3. Select **{BLOCK, Edit/translate, Detab}**. All tabs will be converted to the correct number of spaces.

Notes:

To prevent tab characters from being inserted into your file by the **<Tab>** key and some columnar operations, set **{CONFIG, Tab/Fill, Expand <Tab> with spaces}** to "7".

These functions are implemented by the macros **`detab.vdm`** and **`retab.vdm`**. Users experienced with the VEDIT PLUS macro language can change them to better suit their needs.

Block Delete

Delete the highlighted block of text.

Keystroke Equivalent:

(None). However, **[DELETE]** also deletes highlighted blocks.

Full Description:

A stream, columnar or line block of any size can be deleted.

➢ **To delete a block of text:**

1. Highlight the desired text. You can highlight a "stream", "columnar" or "line" block of any size.
2. Select **{BLOCK, Edit/translate, Block delete}**. Or press **[DELETE]**.

Notes:

It is usually simpler to delete a highlighted block by pressing **[DELETE]**. However if **{CONFIG, Emulation, Special emulation modes}** is configured to not allow **[DELETE]** to delete blocks, this item must then be used.

If the block being deleted is too large for "Undo" to restore it, you will be prompted for verification whether to still delete the block.

You can also effectively "delete" a (small) block of text by moving it into a text register.

See Also:

{EDIT, Delete, Delete}, **{EDIT, Undo, Deletion}**
{BLOCK, Set stream marker}, **{BLOCK, Move to register}**
[DELETE]

Block Fill

Fill (overwrite) the marked block with any character, usually spaces.

Keystroke Equivalent: (None)

Full Description:

The marked block is filled (overwritten) with the configurable "block-fill" character, typically spaces. A stream, columnar or line block of any size can be filled.

This item is primarily used to fill (erase) columnar blocks without affecting the position of other columns.

See Also:

{CONFIG, Tab/Fill, Block fill character}

Insert Empty Column

Insert an empty columnar block between other columns.

Keystroke Equivalent: (None)

Full Description:

This item inserts an empty block, consisting of the configurable "block-fill" character, at the cursor position. A block is first marked (highlighted) to indicate the size and position of the empty block to be inserted. The existing marked text is then shifted (indented) to the right.

This item makes it easy to increase the amount of space between two columns of text.

➢ **To insert an empty columnar block:**

1. Move the cursor to the first line and column at which you want the empty block inserted. Select **{BLOCK, Set column marker}** (normal: **<Alt-I>**).
2. Move the cursor to the bottom right corner for the empty block. Existing text will probably be highlighted. Select **{BLOCK, Set column marker}** again.
3. Select **{BLOCK, Edit/translate, Insert empty column}**. An empty columnar block will be inserted and any existing text on the selected lines will be indented to the right.

HINT: This useful (and unique) function is best understood by trying it in a small test file.

Notes:

This function may seem confusing at first because the highlighted text is not really processed; the highlighting simply indicates the size and position of the empty block being inserted. The highlighted text and the text to the right of it will be indented to make room for the empty block.

You can reduce the space between two columns of text, e.g. newspaper style, by highlighting one or more columns of spaces and then deleting them. "Insert empty column" performs the opposite operation of increasing the space between two columns of text.

This function is most useful with columnar blocks, but it could be used with stream and line blocks too.

Strip High bit

Strip the "high" (8th bit) from all characters in the marked block.

Keystroke Equivalent: (None)

Full Description:

Some word processors, such as WordStar (tm), set the high bit on some text characters for internal formatting purposes. These text files are then difficult to edit in VEDIT because the high bit characters appear as graphics characters and the words are not readable.

This function strips the high (8th) bit from all characters in the marked (highlighted) block. **{BLOCK, Select all}** can be used to mark the entire file as a block.

Notes:

Use this function with care! It will convert IBM PC graphics characters and non-english language characters into meaningless text characters.

See Also:

"Screen Display and Keyboard Characters" in Chapter 4.

Sort Lines

Sort the selected lines according to the marked columns (field).

Keystroke Equivalent: (None)

Full Description:

This function sorts lines (records) into ascending order. You must first highlight a columnar block consisting of the columns (field) to be used as the "sort key". For example, it could be the Last-name or Zipcode in a database. The columnar block must extend from the first line to be sorted through the last line to be sorted.

➢ **To sort all lines in a block or entire file:**

1. Position the cursor on the first line to be sorted and in the left-most column of the field to be sorted.

   Select **{BLOCK, Set column marker}** (**<Alt-I>**) to begin a columnar block.

2. Position the cursor on the last line to be sorted and in the right-most column of the field to be sorted. To sort an entire file, this must be the last line of the file.

   Select **{BLOCK, Set column marker}** (**<Alt-I>**) again.

3. Select **{BLOCK, Edit/translate, Sort lines}**. The lines will be sorted and VEDIT will display its progress.

Notes:

By first sorting according to secondary keys and then sorting according to the primary key, you can sort according to multiple keys.

This menu item can only be selected after you have marked a columnar block.

See Also:

"Sorting Lines in a Block / File" in Chapter 4.
"SORT.VDM Sort Macro" in Chapter 5.

Translate from EBCDIC

Translate to EBCDIC

Translate a block to/from EBCDIC or use a custom translation table.

Keystroke Equivalent: (None)

Full Description:

A marked block or entire file can be translated using the built-in EBCDIC table, the supplied ANSI table or with a user-created table. When a block/file is translated, each byte is simply converted to another byte according to the current table; the size of the file does not change.

➢ **Translate a file from EBCDIC to ASCII:**

1. Open the EBCDIC file in the normal manner, e.g. with **{FILE, Open}**.
2. Select **{BLOCK, Select all}** to mark the entire file as a block.
3. Select **{BLOCK, Edit/Translate, Translate from EBCDIC}** to translate the file to ASCII.

   You may need to change **{CONFIG, File handling, File type}** to "0" or "1" to recognize ASCII "newlines" and make the file more readable.
4. Select **{FILE, Close buffer}** or **{FILE, Exit}** to save the translated file.

Similarly, an ASCII file can be translated to EBCDIC with "Translate to EBCDIC".

Notes:

The "EBCDIC" displayed in the menu will change when a custom translation table is loaded.

You can also display an EBCDIC file in normal ASCII without having to translate the file. See **{VIEW, Toggle display mode}**.

The topic "Translating a Block or File" in Chapter 4 describes VEDIT's translation capabilities in more detail.

EBCDIC conversion packages are available for translating EBCDIC files with packed (signed) decimal, packed binary, zoned and other special COBOL fields into ASCII. Please refer to the "EBCDIC" page on our Web site or contact us for details.

Load Translate Table

Load a custom translation table to replace the built-in EBCDIC table.

A custom translation table can be loaded to replace the built-in ASCII-EBCDIC table. ANSI.TBL is supplied for translating between the IBM PC graphics characters and the ANSI (Windows) character set. USER.TBL is supplied as a template for creating your own translation table.

➢ **To load a new translation table:**

1. Select **{BLOCK, Edit/translate, Load translate table}**.
2. You are prompted for the desired filename which typically has a ".tbl" extension.

Each translation table file includes the new name that will replace "EBCDIC" in the **{BLOCK, Edit/translate}** sub-menu.

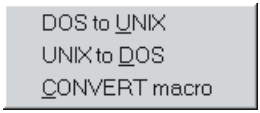
See Also:

{BLOCK, Edit/translate, Translate from EBCDIC}

Convert Newlines (Sub-menu)

Introduction:

This menu converts newline characters to the selected type for DOS/Windows, UNIX or Macintosh. Also converts between fixed-length (database) records and normal text files.



DOS to UNIX

UNIX to DOS

Convert all CR+LF in a block to just LF.

Convert all LF in a block to CR+LF.

Keystroke Equivalent: (None)

Full Description:

"DOS to UNIX" converts all "*newlines*" in the highlighted block from the type for DOS/Windows (CR+LF) to the type for UNIX (LF only). In other words, it replaces all "Carriage-Return and Line-Feed" character pairs with just a Line-Feed character.

If the entire file is highlighted, it converts a DOS/Windows text file into a UNIX text file. It then also changes **{CONFIG, File handling, File type}** to "1".

"UNIX to DOS" converts all "*newlines*" in the highlighted block from the type for UNIX (LF only) to the type for DOS/Windows CR+LF). In other words, it replaces each Line-Feed character with a "Carriage-Return and Line-Feed" character pair.

If the entire file is highlighted, it converts a UNIX text file into a DOS/Windows text file. It then also changes **{CONFIG, File handling, File type}** to "0".

For example, "DOS to UNIX" can be used to fix a block which was pasted from a DOS/Windows file into a UNIX text file.

Notes:

These items runs the **`convert.vdm`** macro without any prompts. However, if there is an error, e.g. no newlines are found, it will return to the CONVERT.VDM menu.

If you accidentally select "UNIX to DOS" with a DOS text file, you will have two Carriage-Returns at the end of each line. This can be corrected by selecting

"DOS to UNIX"; you may also need to manually change **{CONFIG, File handling, File type}** to "0".

To see what is really at the end of each line, select **{VIEW, Toggle display mode}** (**<Alt-D>** or toolbar) several times until the window displays all characters in hexadecimal. A Carriage-Return is hex code "0D", a Line-Feed is hex code "0A".

See Also:

"File Types - Win/DOS, UNIX, Mac, Binary" in Chapter 4.
{BLOCK, Convert newlines, CONVERT macro}

CONVERT macro

Convert to/from fixed-length record files. Convert to/from Macintosh files.

Keystroke Equivalent:

(None)

Full Description:

This item converts all newlines in the highlight block to the selected type for DOS/Windows, UNIX or Macintosh. It can also convert between fixed-length (database) records and normal text files.

For example, it can convert a database file without any newlines into a normal text file with newlines.

This item displays a simple menu of conversion options.

Notes:

This function is implemented by the macro **`convert.vdm`**. Users experienced with the VEDIT PLUS macro language can change it to better suit their needs. For example, you could easily change all **{BLOCK, Convert newline}** functions to convert the entire file if no block if highlighted; as supplied, the macro gives an error if no block is highlighted.

See Also:

"File Types - Win/DOS, UNIX, Mac, Binary" in Chapter 4.

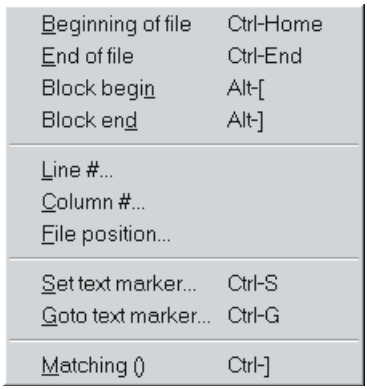
Goto Menu

Keystroke Equivalent:

<Alt-G>, This is a keystroke macro.

Introduction:

The Goto Menu gives 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 operation in this menu, look in Chapter 7 (Edit Function Reference).



Beginning of File

End of File

Position the cursor at the beginning or end of the file.

Keystroke Equivalent:

<Ctrl-Home>, This is a keystroke macro.

<Ctrl-End>, This is a keystroke macro.

Full Description:

"Beginning of file" moves the cursor to the first character in the file.

"End of file" moves the cursor past the last character in the file.

Notes:

These functions may take some time to perform on very large files when the entire file cannot be kept in memory. If the "LI" of the word "LINE" on the status line is capitalized, the beginning of the file is in memory. If the "NE" is capitalized, the end of the file is in memory.

If you only need to browse a huge file, you should open it in "Read-only" mode. This function and other cursor movements will then be performed instantly on even the largest files.

Block-Begin

Block-End

Move the cursor to the beginning or end of the currently highlighted block.

Keystroke Equivalent:

<Alt-[>, this is a keystroke macro.

<Alt-]>, this is a keystroke macro.

Full Description:

These functions move the cursor to the beginning or end of a highlighted block. It is useful when you forget precisely where the beginning of a large block is after it has scrolled off the screen.

Notes:

To restrict a search within a highlighted block you must first highlight the block and then move the cursor to the beginning of the block. **{GOTO, Block-begin}** is a convenient way of doing this.

If a block is extremely large, these functions may result in some disk activity and delay.

See Also:

{BLOCK, Set stream marker}
{SEARCH, Search} - Search within a block

Line

Move the cursor to the beginning of a specific line or record.

Keystroke Equivalent: (None)

Full 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, the line numbers of all following lines will be changed accordingly.

{GOTO, Line #} is useful when working with large files. If you know 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 (record) in the file:**

1. Select **{GOTO, Line #}**.

2. You are prompted for the line number. Enter the desired line number and press **<Enter>**. You can also enter numeric expressions such as "(1245+858)/2".

Notes:

When a file is opened for editing, you can start editing on a particular line number by following the filename with the "**-L***nnn*" option.

This function may take some time to perform on very large files when the entire file cannot be kept in memory.

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)

Full Description:

➢ **To go to a specific column in the current line:**

1. Select **{GOTO, Column #}**.
2. You are prompted 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. 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.

The mouse and bottom scroll bar can also be used to scroll sideways in long lines.

File Position

Move the cursor to a specific file position.

Keystroke Equivalent: (None)

Full Description:

This function moves the cursor directly to the 'n'th character (offset) in a file. The first character in the file is considered to be the 0'th character. The position at the end of the file is the same as the file size. (There are good reasons for counting from 0).

This function is often useful when editing binary and data files.

➢ **To go to a specific file position (offset):**

1. Select **{GOTO, File position}**.
2. You are prompted for the file position. Enter the desired position and press **<Enter>**. You can also enter numeric expressions such as "128000/4", or enter numbers in hexadecimal such as "0xF7A43".

Set Text Marker

Set a text marker at the current cursor location.

Keystroke Equivalent:

<Ctrl-D>, This is a keystroke macro.

Full Description:

Ten invisible "text markers" are available for remembering locations in your file and returning 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 continue to stick to the same character.

Marker information is not saved in files. As a result, markers are maintained only until you close the file or exit VEDIT. However, if you restore a previous edit session, the text markers are also restored.

➢ **To set a text marker:**

1. Move the cursor to the desired position.
2. Select **{GOTO, Set text marker}**. You are prompted with:

   ```
   Marker number: (0-9)
   ```

3. Enter the desired marker number "0" through "9". As a shortcut, you can immediately press **<Enter>**, **<Ctrl-D>** again (or any function/control key) to select marker "0".

Notes:

Text markers are invisible; the only way to determine where, or if, they are set is by using **{GOTO, Goto text marker}** to position the cursor at the marker.

{BLOCK, Remove markers} only removes the block markers, it does not affect the text markers. There is no way (or reason) to remove text markers once they are set.

If you restore a previous edit session, all text markers are also restored.

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. "Undoing" the deletion will not restore the marker to its original position.
- Moving a block that contains a marker does not move the marker. The marker behaves exactly as if the original text were 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.

Full 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, Goto text marker}**.
2. You are prompted for the text marker. Enter the desired marker number "0" through "9". As a shortcut, you can immediately press **<Enter>** or **<Ctrl-G>** again to select marker "0".

Notes:

See the description of **{GOTO, Set text marker}** for a discussion of text markers.

When working with a large multi-megabyte file, **{GOTO, Goto text marker}** may take a few moments to execute.

See Also:

{GOTO, Set text marker}

Matching ( )

Move the cursor to the character which is logically paired with the character at the cursor.

Keystroke Equivalent:

<Ctrl-]>, This is a keystroke macro.

Full 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 the following expression and select this item, the cursor will be moved to the last ")" instead of the first one it finds.

((a/2) + (b/3))

While it is easy to see that all parentheses properly match in the above expression, some expressions, functions and program source code routines 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 the 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 is given.

HINT: Here's a method 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 the other end of the file without an error message, everything is properly paired.

Misc Menu

Keystroke Equivalent:

<Alt-M>, This is a keystroke macro.

ASCII Table

Display ASCII table and optionally enter any desired characters.

Keystroke Equivalent: (None)

Full Description:

"ASCII table" displays a table of all possible characters with their corresponding decimal value. This includes all control and graphic characters.

The ASCII table is particularly useful for entering graphics characters when 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, scroll the window until the desired character is displayed.
2. Highlight the character to be inserted with the mouse or the cursor keys.
3. Press the **[Insert]** button. If desired, additional characters can be inserted.

DOS, UNIX and QNX Versions:

The display is three "windows" long, press **<Enter>** to toggle to the next window. Press "**I**" to insert a particular value — you will be prompted for its decimal value. Only one character can be entered.

Notes:

If you already know the decimal value of a character you can enter it (with Windows or DOS) by holding down the **<Alt>** key and typing the value on the numeric keypad; the character will be entered when you release the **<Alt>** key.

The IBM PC displays the characters with values 00 and 255 as a space, effectively making them invisible. For this reason, VEDIT can optionally display the "Null" (value 00) as any other character; this is set with **{CONFIG, Characters/Cursors, Null display character}**.

VEDIT can display control and graphics characters in the file in several ways. This is described under "Screen Display & Keyboard Characters - Control and Graphics Character display" in Chapter 4 (Editing Guide).

You can also insert control characters using **[ENTER CTRL]**.

See Also:

"Entering Control and Graphics Characters" in Chapter 4.
[ENTER CTRL]

Box Drawing Mode

Draw "graphic" boxes using cursor keys.

Keystroke Equivalent: (None)

Full Description:

Enters a special mode for drawing decorative boxes in the current file using the cursor keys. The style for vertical and horizontal lines is selected with **{CONFIG, Misc, Box drawing style}**.

Press **<Esc>** to exit this mode and return to normal editing.

Notes:

This function is only available when a font using the "OEM" character set has been loaded. "ANSI" fonts do not have box drawing characters.

This function is implemented by the **`box-draw.vdm`** macro.

DOS Shell

Suspend VEDIT and display the DOS prompt.

Keystroke Equivalent: (None)

Full Description:

"DOS Shell" temporarily suspends VEDIT and "shells out" to a DOS box. You can then run any DOS command and/or programs. These commands can range from simple directory and file management commands (i.e. "DIR", "COPY", "DEL", "CHDIR") to running other application programs.

Give the DOS command "**exit**" to return to VEDIT.

{MISC, DOS 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 temporarily suspend VEDIT and enter DOS:**

1. It is a good idea to select **{FILE, Save all}** to save all file before shelling to DOS.
2. Select **{MISC, DOS shell}**. You may now see a copyright notice for your operating system and the DOS command prompt.
3. You can now execute DOS commands and other software.
4. To return to VEDIT, type "**exit**" at the DOS prompt.

WARNING: If you forget to return to VEDIT before you shut down 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 files before shelling out to DOS.

DO NOT delete files that are currently open for editing or have a filename extension of "***.r***$$" or "***.r***R$". These may be temporary files in use by VEDIT.

DOS Version:

While shelled out to DOS, VEDIT changes the DOS prompt to "*pathname*>>". The double ">>" is a reminder that VEDIT is still loaded. This can be changed with **Config_String(OS_PROMPT)** in the **`vedit.cfg`** file.

YOU SHOULD NOT load any "resident" (TSR) programs while shelled out to DOS!

If V-SWAP is installed in memory, VEDIT will use it to swap itself out of memory before re-entering DOS. V-SWAP can be disabled by setting **{CONFIG, File handling, Use V-SWAP when entering DOS}** to "No".

If VEDIT has used up all available memory and V-SWAP is not installed or disabled, you will not be able to shell out to DOS.

Notes:

The Windows 95/98 version shells to DOS with the MSDOS95.PIF file in the *VEDIT Home Directory*. If desired, right-click on it, select "Properties" and change whether the DOS window is opened full-screen or the size of the window.

{MISC, DOS shell} will not work unless the DOS "COMSPEC" parameter is properly set. This is usually set automatically or it may have an override in your AUTOEXEC.BAT file using a command similar to the following:

```
SET COMSPEC=C:\COMMAND.COM
```

Refer to a DOS manual for information on the "COMSPEC" parameter.

See Also:

"RUNSHELL - Run other programs" in Chapter 5.
{MISC, Run program}

Run Program

Execute a Windows/DOS program and return to VEDIT.

Keystroke Equivalent: (None)

Full Description:

"Run program" executes DOS commands or runs Windows/DOS programs. Unlike **{MISC, DOS shell}**, **{MISC, Run program}** automatically returns to VEDIT as soon as the requested command or program is completed. Use **{MISC, DOS shell}** to run multiple commands or programs and use **{MISC, Run program}** to run a single command or program.

➢ **To execute a single DOS command or program:**

1. Select **{MISC, Run program}**. You are prompted with:

   `DOS command / program:`

2. Enter the desired DOS command or program name. You may enter any valid DOS command or run any program which may be run using a single command.

Notes:

The functions **{Save and run program 1/2}** in the default **{USER}** menu are often preferable. They have the advantage of saving and closing all files before running another program; when done, the files are re-opened.

All Notes and Warnings for **{MISC, DOS shell}** also apply here.

If you frequently run a program which requires a series of commands to execute, you should consider creating a "batch" file and using **{MISC, Run program}** to execute the batch file.

See Also:

"RUNSHELL - Run other programs" in Chapter 5.
{MISC, DOS shell}

WILDFILE Macro

Start up the multiple file processing macro WILDFILE.VDM.

Keystroke Equivalent: (None)

Full Description:

This starts up the **`wildfile.vdm`** macro which is fully described in Chapter 5 (Advanced Topics).

This macro is often used to perform a search and/or replace on entire groups of files. This can include files currently being edited and/or any other files.

The groups of files to be processed can be specified using the wildcard characters "*" and "?". For example, you could perform a search and/or replace on all .c and .h files in any directory or directories. Optionally, all matching files in all subdirectories can also be processed.

See Also:

"WILDFILE - Multi-file Processing" in Chapter 5.
{MISC, Load/Execute macro}

More Macros

Load and execute other supplied macros.

Keystroke Equivalent: (None)

Full Description:

This sub-menu quickly loads and executes many of the macros supplied with VEDIT. Many of these macros are described in detail in Chapter 5 (Advanced Topics).

Select this sub-menu and press **[HELP]** (**<F1>**) for a short description of each macro.

Load/Execute Macro

Load/Execute User Macro

Load and execute a command macro.

Keystroke Equivalent:

<Ctrl-F7>, This is a keystroke macro.

Full Description:

These items loads and immediately executes a VEDIT "command macro". It is typically used to load and execute macros which are not listed in **{MISC, More macros}**, such as macros which you write yourself.

Most macros supplied with VEDIT are installed into the *VEDIT Macro Directory*, typically **c:\vedit\macros**. We suggest that user written macros be saved in the *VEDIT User Macro Directory*, typically **c:\vedit\user-mac**.

➢ **To load and execute a command macro:**

1. Select **{MISC, Load/Execute macro}**.
2. Either enter the desired filename or use the point and shoot filename selection.

   "Load/Execute Macro" defaults to all ".vdm" files in the *VEDIT Macro Directory*, e.g. **c:\vedit\macros*.vdm**.

   "Load/Execute User Macro" defaults to all ".vdm" files in the *VEDIT User Macro Directory*, e.g. **c:\vedit\user-mac*.vdm**. We suggest that you save your own macros in this directory.
3. You are prompted for the text register number in which to load the macro. By convention, macros are usually loaded into registers 10 through 100. The default register is "100".

DOS, UNIX and QNX Versions:

If "terse" dialog boxes have been selected, press **<Enter>** at the short filename prompt for point and shoot file selection.

Notes:

When entering a command macro filename, you can leave off the default ".vdm" extension.

Don't confuse "command macros" stored in files and text registers with "keystroke macros". Command macros are programs written in the VEDIT PLUS macro language. Keystroke macros are "strings" of key-presses, possibly including macro language commands, that are assigned to "hot-keys".

Use **{HELP, Text registers}** to determine which registers are empty and therefore available for holding command macros.

If you use a custom macro often, you may want to add it to the **{USER}** menu for easier access.

See Also:

"Command Macros" in Chapter 5.
{HELP, Text registers}, **{MISC, Execute macro}**

Execute Macro

Execute the command macro stored in a text register.

Keystroke Equivalent:

<Ctrl-F8>, This is a keystroke macro.

Full 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 used to repeatedly execute a macro after it is loaded and executed for the first time with **{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>**). Then enter a text register number of "***b*+buffer**" where '*b*' is the buffer number.

See Also:

{MISC, Load/Execute macro}

Load Template File

Load a template editing macro and enable template editing

Keystroke Equivalent: (None)

Full Description:

This item loads a new template editing macro file, such as **`c.vtm`**, **`html.vtm`** or a custom macro. It also enables **{CONFIG, Programming, Enable template editing}** in the current file.

You are prompted for the desired filename. The file selection dialog box starts with all **`*.vtm`** files in the *VEDIT Home Directory*.

Notes:

VEDIT can automatically load a template editing macro during startup, and enable this feature for all .c and .html files being edited. See the topics "Template Editing" and "STARTUP.VDM File" in Chapter 5 for details.

{CONFIG, Programming, Enable template editing} is only enabled in the current file (buffer) regardless of how **{CONFIG, Config all buffers}** is set. You may have to manually enable it for other buffers.

Load Syntax File

Load a syntax highlighting definition file and enable this feature

Keystroke Equivalent: (None)

Full Description:

This item loads a new syntax highlighting definition file, such as **`c.syn`**, **`html.syn`** or a custom file. It also enables **{CONFIG, Programming, Enable color syntax highlighting}** in the current file.

You are prompted for the desired filename. The file selection dialog box starts with all **`*.syn`** files in the *VEDIT Home Directory*.

Notes:

VEDIT can automatically load a syntax highlighting file for C or HTML during startup, and enable this feature for all **`.c`** or **`.html`** files being edited. See the topics "Syntax Highlighting" and "STARTUP.VDM File" in Chapter 5 for details.

{CONFIG, Programming, Enable color syntax highlighting} is only enabled in the current file (buffer) regardless of how **{CONFIG, Config all buffers}** is set. You may have to manually enable it for other buffers.

Load Compiler Support

Load the compiler support items into the **{Tools}** menu.

Keystroke Equivalent: (None)

Full Description:

This item loads the compiler support items into the **{Tools}** menu, replacing any existing **{Tools}** or **{Tutorial}** menu.

You can load either the "normal" compiler support which uses the **`compile.cnf`** configuration file or the Java SDK specific support which uses the **`java-sdk.cnf`** configuration file and a slightly different **{JavaTools}** menu. You can easily switch between Java SDK and another compiler by selecting **{MISC, Load compiler support}** again.

Notes:

If the normal compiler support is selected, it loads the **`compile.mnu`** menu file. If the Java SDK compiler support is selected, it loads the **`java-sdk.mnu`** menu file.

For the compiler support to work under DOS, it is usually necessary for V-SWAP to already be installed in memory.

This function is implemented by the **`compile.vdm`** macro.

The original **`startup.vdm`** macro loads the **{Tutorial}** menu, but can easily be modified to load the **{Tools}** menu instead.

See Also:

"Integrated Compiler Support" in Chapter 5.
On-line help topic "Compiler Support" (DOS: "COMPILE") .

Load {TOOLS} Menu

Load {USER} Menu

Load a new **{TOOLS}** or **{USER}** menu.

Keystroke Equivalent: (None)

Full Description:

These items load a new menu file as the **{TOOLS}** or **{USER}** menu. You are prompted for the desired filename. The file selection dialog box initially displays all **`*.mnu`** files in the *VEDIT Home Directory*.

Since only **`user.mnu`** is supplied with VEDIT and is automatically loaded during startup, this function's purpose is to load custom **{USER}** menus.

Notes:

The topic "{USER} and {TOOL} Menu" in Chapter 5 describes how to modify the existing **{USER}** menu or create a custom ".mnu" file.

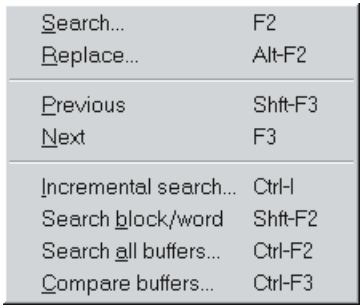
Search Menu

Keystroke Equivalent:

<Alt-S>, This is a keystroke macro.

Introduction:

You can easily and quickly search for a particular string of text and optionally replace it with different text. Just the current file or all open files can be searched. Any two open files can also be compared.



Search

Search for a string of text.

Keystroke Equivalent:

<F2> or **<Ctrl-F>**, These are keystroke macro

Full Description:

Starts a new search in the current file for the specified *search string*. The flexible search dialog box includes options for searching only for entire words, making the search case sensitive, or restricting the search to the highlighted block.

Besides searching for a literal sequence of characters, you can also search using VEDIT's powerful *pattern matching* or *regular expressions*. These search modes are fully described under "Search and Replace" in Chapter 4 (Editing Guide). Pattern matching is the default.

You can search for additional occurrences in the forward or backward directions. You can also search for additional occurrences in other files.

➢ **To perform a search:**

1. Select **{SEARCH, Search}** (**<F2>** or **<Ctrl-F>**).
2. Enter the *search string*, the sequence of characters to be located.
3. Check that the desired search mode is selected, e.g. "Simple", "Pattern matching" or "Regular expressions". See below or use on-line help.

Select any desired search options.

4. Press **<Enter>** or the **[Next]** button to search in the forwards direction from the cursor position to the end of the file.

   Or select **[Prev]** to search backwards towards the beginning of the file.

   If the string is found, the cursor is positioned on its first character and the entire string is highlighted. If the search string is not found, VEDIT displays an error message.

5. To search for the next occurrence of the same string, select **{SEARCH, Next}** (<**F3**>) or **{SEARCH, Previous}** (<**Shft-F3**>).

HINTS: Previous search strings can be recalled. In the Windows version, press the recall arrow. (DOS/UNIX/QNX: press **[CURSOR UP]**). You can also edit the search string being entered or that you recalled.

The on-line help for searching is extensive. It explains how to search for special characters such as "newline", "null", control and graphics characters.

Search Modes:

The dialog box's initial search mode is set by **{CONFIG, Search options, Default search mode}**. The default mode is "Pattern matching".

Simple A simple search without pattern matching or regular expressions is performed. There are no special characters; there is no "wildcard" type searching.

Pattern VEDIT's "pattern matching" can be used to search for many types of characters, e.g. "wildcards". In this mode, only the "|" (pipe or vertical bar) character is special. For example use "|**013**" to search for a carriage return, "|**000**" to search for a null. Use "||" to search for a single "|" in the file.

Reg-Exp UNIX-style "regular expressions" can be used to search for complex patterns of characters; portions of the matched text can also be used in the replacement side. Regular expressions are much more technical than pattern matching. It is generally not suitable for simple searches because many characters have a special meaning. To search for a special character, precede it with "\".

Reg-Exp (Max) Both "minimal" and "maximized" type regular expressions are supported. See "Maximize Regular Expressions" in Chapter 4 for a description of this technical topic.

Search Options

Case The search is case sensitive, e.g. a search for "the" will not match "The". Otherwise the search is not case sensitive. The dialog box's initial value is set by **{CONFIG, Search options, Default Case-sensitive option}**.

Word	Restricts the search to entire "words", e.g. a search for "the" will not find "there", "other" and similar words. It is equivalent to entering, e.g. "|Sthe|S" using pattern matching.
Begin of file	Start searching from the beginning of the file; otherwise the search starts from the current cursor position. In a block search, it starts searching at the beginning of the block.
Local	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 can be useful when searching in huge files to prevent an unsuccessful search from wasting time examining the entire file when you only want to search nearby.
Block	Restricts the search to the highlighted block. This option is selected automatically if the cursor is within, or immediately past the block.

DOS, UNIX and QNX Versions:

The "terse" dialog box only displays some options. Immediately press **<Enter>** to switch to the full dialog box.

In the terse dialog box, the button **[Again]** re-uses the previously entered search string. It is often used in conjunction with the **[ ] B-O-F** option to restart the search from the beginning of the file, or with the **[ ] Prev** option to search again in the backwards direction.

Searching within a block.

You can restrict the search (or replace) to a highlighted block, even a columnar block. This is 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 as a stream, line or 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, we suggest moving the cursor to the beginning of the block, e.g. with **{GOTO, Block-begin}** (**<Alt-[>**). The search starts at the cursor position, not necessarily at the beginning of the block. See the rules below.
3. Select **{SEARCH, Search}** (**<F2>**).

   If the cursor is within the block, the **[ ] Block** option will already be selected; otherwise select this option.

   Enter the search string and select any options as usual. Press the button corresponding to the desired search direction.

Many other editors simply force the search to the beginning of the block after you highlight the text. However, VEDIT has 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}** or **{SEARCH, Replace}** the **[ ] Block** search option is automatically selected. Disable it if you want to search the entire file.

2. The search normally starts at the current cursor position and not automatically at the beginning of the block.

   However, if the cursor is not within the block and you select the **[ ] Block** option, a forward search automatically starts at the beginning of the block. Similarly, a backwards search starts at the end of the block.

3. The entire search string must be found within the highlighted text; this 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:

The search and replacement strings are limited to 260 characters.

The contents of a text register can be used as the entire search/replace string or as a portion of it. In pattern matching mode, use "**|@(***r***)**" to include text register '*r*' in the search/replace string. In regular expression mode, use "**\@(***r***)**". This permits "variable" search/replace strings.

To search for a "newline", use "**|N**" in pattern matching mode, or "**\N**" in regular expression mode. In the DOS version, you can alternatively press **<Ctrl-N>**. This will display the "newline" characters according to the current file type, e.g. **<CR><LF>**. (The **<Ctrl-N>** is not configurable; think "N" for "newline".)

To search for control (or other) characters, use "***\|ddd***" in pattern matching mode, or "**\d***ddd*" in regular expression mode, where '*ddd*' is the decimal value of the character. In the DOS version, you can alternatively type **[ENTER CTRL]** (**<Ctrl-Q>**) followed by the control character.

Character values can also be entered in hexadecimal using "**|H***hh*" in pattern matching mode, or "**\h***hh*" in regular expression mode.

To search for a null character with decimal value "000", use "***\|000***" in pattern matching mode, or "**\d***000*" or "**\0**" in regular expression mode.

DOS and Windows only: To search for graphics (or other) characters, hold down the **<Alt>** key while you type the decimal value of the character on the numeric keypad. In Windows, precede the decimal value with "**0**" to prevent Windows from translating the character from the IBM-PC (OEM) to the ANSI character sets; e.g. to search for the character with value 132, type **<Alt> 0132**.

See Also:

"Search and Replace" in Chapter 4.
{SEARCH, Again}

Replace

Locate a text string and replace it with another text string.

Keystroke Equivalent:

<Alt-F2> or **<Ctrl-H>**, They are keystroke macros.

Full Description:

Starts a new search for the specified *search string*; and if matching text is found, replaces it with the *replace string*.

The replacement can be performed a single time, globally throughout the file or selectively.

The same search mode and options as for **{SEARCH, Search}** are used.

➢ **To perform a search and replace operation:**

1. Select **{SEARCH, Replace}** (**<Alt-F2>**).
2. Enter the "search string" in the same way as for **{SEARCH, Search}**.
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 found, the cursor will be positioned just past the first occurrence of the located text and you are prompted with:

Select the desired option:

[Yes] Replace the text for this occurrence and immediately search for the next occurrence.

[No] Don't replace the text for this occurrence; immediately search for the next occurrence.

[All] Replace the text for this and all remaining occurrences without prompting. (Note: "all" occurrences are replaced only if the operation was started at the beginning of the file.)

[One] Replace the text for this occurrence and return to normal editing.

[Cancel] Don't replace the text for this occurrence and cancel the replace operation; return to normal editing.

Notes:

The description and notes for **{SEARCH, Search}** apply here too.

With VEDIT's normal "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]}ello}" 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 4 (Editing Guide).

See Also:

{SEARCH, Search}, **{SEARCH, Again}**

Next

Previous

Repeat the last "Search" or "Replace" operation again in the forwards or backwards direction.

Keystroke Equivalent:

<F3>, This is a keystroke macro for **{SEARCH, Next}**.

<Shft-F3>, This is a keystroke macro for **{SEARCH, Previous}**.

Full Description:

"Next" repeats the last "Search" or "Replace" operation again in the forwards direction toward the end of the file, starting at the cursor position in the current file.

Similarly, "Previous" repeats it in the backwards direction toward the beginning of the file.

The same search options will be used that were 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.

If desired, you can switch to another buffer (file) and repeat the search/replace there.

Incremental Search

Search for text as you enter the search string.

Keystroke Equivalent:

<Ctrl-I>, This is a keystroke macro.

Full Description:

This items prompts for a search string and searches the current file, starting from the cursor position, for the accumulated search string after each keystroke. Press **<Esc>** when done. This is a convenient way to search without having to enter more of the search string than is necessary.

Following each keystroke, if matching text is found, it is highlighted. Otherwise the previously marked text remains highlighted. If desired, you can then press **<Backspace>** and change the search string.

The following function/control keys can be used:

<Esc>	Finishes the incremental search.
<Backspace>	Erases the last entered character and backs up to the previous cursor position.
<Ctrl-N>	Enters the current newline character(s) into the search string.
[ENTER CTRL]	(Default: **<Ctrl-Q>**) Enters the subsequent control character into the search string.
<Enter>	Searches for the next occurrence.

You can later select **{SEARCH, Next}** or **{SEARCH, Previous}** to find the next/previous occurrence of the string.

Notes:

The search is case sensitive if **{CONFIG, Search options, Default case sensitive option}** is enabled. The search mode (simple, pattern matching, regular expressions) is set by **{CONFIG, Search options, Default search mode}**.

This function is implemented by the **`srchincr.vdm`** macro.

Since incremental searching that fails in huge files can be time consuming, this macro can be set up to initially perform a "local" search. See **`srchincr.vdm`** for details.

Search Block/Word

Search for occurrences of block or word at cursor.

Keystroke Equivalent:

<Shft-F2>, This is a keystroke macro.

Full Description:

This item searches for the next occurrence of the text which is currently highlighted as a block. The search starts at the cursor position. If no block is highlighted, it searches for the word at the cursor position; if the cursor is on a space following a word, it searches for the preceding word and a space.

You can either press the hot-key again or **{SEARCH, Next}** (**<F3>**) to search for additional occurrences.

Notes:

The search is case sensitive if **{CONFIG, Search options, Default case sensitive option}** is enabled.

This function is implemented by the **`srchblck.vdm`** macro.

Search All Buffers

Search all open files for a string of text.

Keystroke Equivalent:

<Ctrl-F2>, This is a keystroke macro.

Full Description:

This items prompts for a search string and then searches all buffers (files) currently being edited for the specified string. You can then make any desired editing changes.

For example, you can easily search all modules (files) that make up a complex program, or all chapters that make up a manuscript.

The search starts with the current buffer and continues with all open buffers. Assuming the buffer has an open file, the entire file is searched. Pressing **[VISUAL EXIT]** (default: **<Ctrl-E>**) searches for the next occurrence, switching to another buffer (file) if necessary. Pressing **[VISUAL ESCAPE]** (default: **<Alt-F10>**) ends the search and restores the normal operation of **[VISUAL EXIT]**.

➢ **To search all open files for a string of text:**

1. Select **{SEARCH, Search all buffers}**.
2. At the prompt, enter the desired search string. Simply press **<Enter>** to use the same string as for the last **{SEARCH, Search}**.
3. When the string is found, you can make any desired editing changes.
4. Press **[VISUAL EXIT]** (**<Ctrl-E>**) to search for the next occurrence.
5. Press **[VISUAL ESCAPE]** (**<Alt-F10>**) when done searching.

Notes:

The search is case sensitive if **{CONFIG, Search options, Default case sensitive option}** is enabled. The search mode is set by **{CONFIG, Search options, Default search mode}**. Other search options cannot be specified. (This function will be an option in the search dialog box in a future version.)

This function is implemented by the **`sallbuff.vdm`** macro.

The WILDFILE macro can also search/replace numerous files without having to simultaneously load them into VEDIT. WILDFILE can search/replace thousands of files in one operation, e.g. it can search all files in a subdirectory tree.

See Also:

"WILDFILE - Multi-file Processing" in Chapter 5.
{SEARCH, Search}

Compare Buffers

Perform a byte-by-byte comparison between two buffers (files).

Keystroke Equivalent:

<Ctrl-F3>, This is a keystroke macro.

Full Description:

This function performs a quick comparison of two buffers (files). It compares the current buffer with the selected one. The comparison starts at the current cursor position of both buffers, and both cursor positions are advanced over all matching characters. To continue the comparison, you must first manually re-align the cursor in both buffers.

As a convenience, if only two files are being edited, this function skips the selection prompt and immediately compares against the "other" file.

➢ **To compare two buffers (files):**

1. If necessary, switch to both buffers and ensure that the cursor in each buffer is in the correct position for starting the comparison. Remain in either buffer.
2. Select **{SEARCH, Compare buffers}** (**<Ctrl-F3>**). If only two buffers are open, the comparison starts immediately and you should notice that the cursor has moved over all matching characters.

   If more than two buffers are open, you are prompted for the "other" buffer number with the buffer selection dialog box.

   ```
   Buffer number:
   ```

3. Enter the desired buffer number and select a buffer using point and shoot. Simply press **<Enter>** or any function/control keys to select the main edit buffer #1.

Notes:

(DOS, UNIX, QNX versions) If "terse" dialog boxes have been selected, erase the default "1" buffer and then press **<Enter>** for point and shoot buffer selection.

To perform a case sensitive comparison, e.g. to compare two binary files, enable **{SEARCH, Config, Default Case-sensitive option}**.

The supplied COMPARE file comparison macro is more sophisticated and often easier to use. It can automatically re-align the files being compared to continue the comparison.

See Also:

"COMPARE - Compare Files" in Chapter 5.

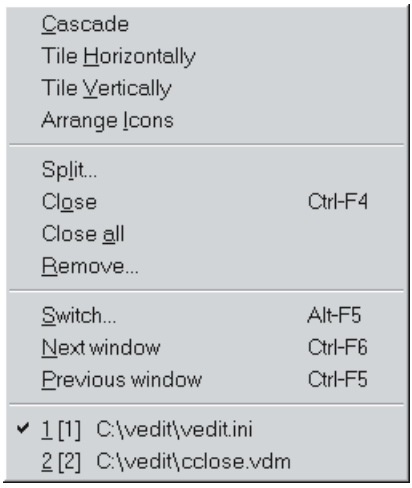
Window Menu

Keystroke Equivalent:

<**Alt-W**>, This is a keystroke macro.

Introduction:

The **{Window}** menu lets you cascade or tile windows, create new (split) windows, close and remove windows, and switch to any desired window.



Cascade

Tile Horizontally

Tile Vertically

Rearrange the existing windows into a cascading or tiled view.

Keystroke Equivalent: (None)

Full Description:

"Cascade" resizes all windows to the same size; it moves them so that they partially overlap each other with all filenames on the top border visible. The most recently used windows will be on top.

"Tile" moves and resizes all windows so that they fit on the screen without overlapping (the windows may be very small). "Tile horizontally" gives a preference to full width windows, "Tile vertically" gives a preference to side-by-side full height windows.

Notes:

Some VEDIT macros create special “reserved” windows at the top or bottom of the screen that cannot be cascaded, tiled, or resized.

See Also:

“Windows” in Chapter 4.

Arrange Icons

Line up minimized icons at bottom of VEDIT’s desktop

Keystroke Equivalent: (None)

Full Description:

“Arrange Icons” neatly arranges all icons corresponding to minimized editing windows at the bottom of the VEDIT desktop area.

Notes:

The minimized icons are not visible if the current window is zoomed to full-size or if any window is otherwise overlaying the icons. You may have to move windows to see them all.

Split

Split the current window into two horizontal or vertical custom sized windows.

Keystroke Equivalent: (None)

Full Description:

This item split the current window into two windows, each displaying the same file. This lets you view and edit two regions 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.

The dialog box lets you specify the location of the new window in the top, bottom, left or right half of the current window. You can also specify the number of lines/columns in the new window. The default value of “0” splits the window exactly in half.

Use **{WINDOW, Next window}** (**<Ctrl-F6>**) **{WINDOW, Previous window}** (**<Ctrl-F5>**), **{WINDOW, Switch}** (**<Alt-F5>**) or the mouse to switch to the new window. You can then examine or edit another part of the same file. Each window on the same file can be scrolled independently. 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 display the same region of the file, you will notice your edit changes occurring in both windows.

You can resize the windows after performing a vertical/horizontal split.

VEDIT supports separate colors for each window. **{MISC, Color toggle}** toggles the current window through different colors.

{FILE, Open (More), Horizontal window} internally performs a "Horizontal split" as it opens a new file for editing. Similarly, **{FILE, Open (More), Vertical window}** performs a "Vertical split".

VEDIT automatically assigns each new window the first unused ID number. However, for routine editing it is not necessary to know a window's ID number.

See Also:

"Windows" in Chapter 4.
{WINDOW, Switch}, **{VIEW, Toggle Hex-mode split}**

Close

Close the current window and close the buffer (file) displayed in it.

Keystroke Equivalent:

<Ctrl-F4>, This is a keystroke macro.

Full Description:

This item closes the current window; it is the same as clicking the mouse on the windows's "close" button. If the window contains an altered file ***AND*** is the only window displaying that file, it closes the file, same as **{FILE, Close}**, prompting whether the file should be saved or abandoned. Otherwise, it only closes the window.

When a buffer (file) is displayed in two or more windows, this function closes (deletes) the additional windows in which it is displayed.

Notes:

We recommend using this function, or (equivalent) clicking the mouse on the windows "close" button, as the normal way to close files.

See Also:

"Editing Multiple Files - Closing Files" in Chapter 4.
{FILE, Close}

Close all (Windows version only)

Closes all windows and buffers (files).

Keystroke Equivalent: (None)

Full Description:

This item closes all windows and buffers; you can save or abandon each modified file, similar to **{FILE, Exit}**.

It should immediately be followed by **{FILE, Open}** (or toolbar) to open additional files. Until a window is opened, VEDIT will not respond to any keystrokes.

See Also:

{FILE, Exit}

Remove

Removes (deletes) the specified window.

Keystroke Equivalent:

(None)

Full Description:

This item prompts for the ID number of the window to be removed. It only closes the window; it does not close the associated buffer (file), if any. This is sometimes referred to as "deleting" the window. Since windows and buffers are independent in VEDIT, this function ***DOES NOT*** affect any files!

Its main purpose is to close multiple (extra) windows per file created with the window split functions, or delete special "command mode" windows created by a "command macro".

If you remove the only window attached to a buffer (file), VEDIT automatically creates a new window when you switch to that buffer. The current window cannot be removed. (You may notice a flicker as the current window is deleted, but then immediately recreated.)

Notes:

VEDIT for Windows users are unlikely to use this (unusual) function during routine editing.

To remove all extra windows, select **{View, Reset}**.

It is usually easier to delete extra windows by simply clicking the mouse on their "close" button.

See Also:

{VIEW, Reset}

Switch

Switch directly to another window.

Keystroke Equivalent:

<Alt-F5>, This is a keystroke macro.

Full Description:

"Switch" prompts for the ID number of the desired window and then switches to the specified window. If the specified window does not exist, the command is ignored. The window can also be selected via point and shoot.

If the new window contains a different file, it also switches to that file (buffer), i.e. it also performs **{FILE, Buffer switch}**. If the new window contains the same file, the editing position is moved to the cursor position in the new window. This makes it easy to view and edit two or more regions of one file.

Notes:

{WINDOW, Switch} is not the same as **{FILE, Buffer switch}**. It is primarily used when one buffer (file) is being edited in two or more windows, e.g. following **{VIEW, Toggle Hex-mode split}**.

You can also switch to a window by clicking the mouse anywhere in the desired window. However, this function can be used to switch to another window when windows are zoomed.

It is usually easier to switch to any desired window by toggling to it with **{WINDOW, Next window}** (**<Ctrl-F6>**) or **{WINDOW, Previous window}** (**<Ctrl-F5>**).

When switching to another window on the same file, there may be some delay while VEDIT brings the new region of the file into memory.

See Also:

{FILE, Buffer switch}
{WINDOW, Next window}.

Next Window

Previous Window

Toggle round-robin style to the next/previous window and its file.

Keystroke Equivalent:

<Ctrl-F6>, This is a keystroke macro.

<Ctrl-F5>, This is a keystroke macro.

Full Description:

"Next window" switches, round-robin style, to the next existing window. Similarly, "Previous window" switches to the previous window. If there is only one window, they have no effect.

If the new window contains a different file, it also switches to that file (buffer), i.e. it also performs **{FILE, Buffer switch}**. If the new window contains the same file, the editing position is moved to the cursor position in the new window. This makes it easy to view and edit two or more regions of one file.

Notes:

You can also switch directly to another window with **{WINDOW, Switch}**.

See the notes for **{WINDOW, Switch}**.

See Also:

{FILE, Next buffer}, **{WINDOW, Switch}**.

Config Menu

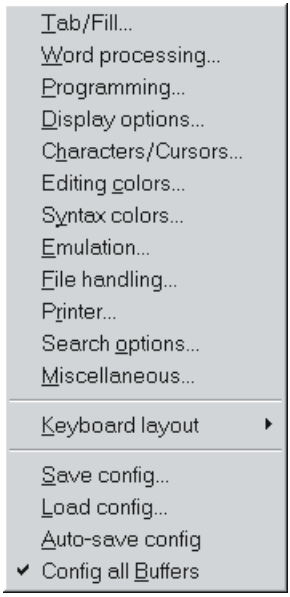
Keystroke Equivalent:

<**Alt-C**>, This is a keystroke macro.

Introduction:

The Config selections let you tailor VEDIT to your applications and personal preferences. Some less commonly used configuration options cannot be configured from inside VEDIT, but only by changing the **`vedit.cfg`** file as described in Chapter 8 (Configuration).

Parameters marked with "(*)" are buffer dependent - you can have a different value for each buffer (file). **{CONFIG, Config all buffers}** explains this in detail.



DOS: ***UNIX:*** ***QNX:*** When you browse through the Config Menu using **[CURSOR UP]** and **[CURSOR DOWN]**, you will see the contents of each sub-menu displayed. This lets you see settings available in each sub-menu before you actually select it. (This assumes **{CONFIG, Display options, Enable sub-menu preview}** is set.)

Windows and DOS versions:

By default, all configuration changes (except for the keyboard layout) are automatically saved for the next time you run VEDIT. Experienced users may want to turn off this feature with **{CONFIG, Auto-save config}** so that configuration changes are temporary unless you explicitly select **{CONFIG, Save config}**.

UNIX and QNX versions:

Any configuration changes you make are temporary and are lost when you exit, unless you explicitly save them to disk with **{CONFIG, Save config}**.

Notes:

When restoring a previous edit session, all configuration settings are restored, regardless of whether you made them permanent.

The default settings for VEDIT are the result of years of experience. We suggest that you at least try our settings before making numerous changes.

Save Config

Save current configuration settings in **`vedit.cfg`** or another file.

Keystroke Equivalent: (None)

Full Description:

This item saves all current configuration settings into the **`vedit.cfg`** file, or other specified file. By saving into **`vedit.cfg`**, the configuration changes will be permanent (or until changed again).

VEDIT automatically configures itself at startup by loading the **`vedit.cfg`** file.

➢ **To make configurations changes permanent by saving into `vedit.cfg`:**

1. Select **{CONFIG, Save config}**. You are prompted with:

   ```
   Filename: C:\VEDIT\VEDIT.CFG
   ```

2. If the default "vedit.cfg" filename is correct, press **<Enter>**. Otherwise edit the path and filename as needed.

Notes:

This function does not save the current keyboard layout. You must use **{CONFIG, Keyboard layout, Save layout}** for this.

`vedit.cfg` is a command macro that contains all of the **Config( )** commands necessary to fully configure VEDIT. Experienced users may prefer to configure VEDIT by directly editing this file as described in Chapter 8.

In the DOS version, you can also save the entire configuration and keyboard layout into the VEDIT.EXE file with **{CONFIG, Misc, Save into VEDIT.EXE}**.

See Also:

"STARTUP.VDM File" in Chapter 5.
{CONFIG, Keyboard layout, Save layout}

Load Config

Configure VEDIT by loading a .CFG configuration file.

Keystroke Equivalent: (None)

Full Description:

You can configure VEDIT by loading a **`.cfg`** file. These files are usually created with **{CONFIG, Save config}**.

➢ **To load a new configuration from disk:**

1. Select **{CONFIG, Load config}**. You are prompted with:

   `Filename: C:\VEDIT\VEDIT.CFG`

2. If the default "vedit.cfg" filename is correct, press **<Enter>**. Otherwise edit the path and filename as needed.

Notes:

VEDIT automatically configures itself at startup by loading the **`vedit.cfg`** file.

This function does not load a new keyboard layout. You must use **{CONFIG, Keyboard layout, Load layout}** for this.

If desired, you can delete most lines from a **`.cfg`** file and keep only those lines that change the default settings.

See Also:

"VEDIT.CFG Config File" in Chapter 8
{CONFIG, Keyboard layout, Load layout}

Auto-save Config

Set up VEDIT to automatically save all configuration changes.

Keystroke Equivalent: (None)

Full Description:

When enabled (default), all configuration changes are automatically saved so that you will have the new configuration the next time you run VEDIT. The configuration changes are saved by immediately updating the **`vedit.cfg`** file.

However, experienced VEDIT users may wish to disable **{CONFIG, Auto-save config}** so that you can make temporary configuration changes. To make configuration changes permanent, you must then select **{CONFIG, Save config}**.

Changes to the keyboard layout are not automatically saved; they can only be saved with **{CONFIG, Keyboard layout, Save layout}**.

Notes:

When restoring a previous edit session, all configuration settings are restored, regardless of whether you made them permanent.

See Also:

{CONFIG, Keyboard layout, Save layout}.

Config All Buffers

Select whether changing any of the "edit buffer dependent" configuration parameters affects all edit buffers or just the current and subsequently created buffers.

Keystroke Equivalent: (None)

Full Description:

For maximum flexibility when editing multiple files, VEDIT maintains a separate set of Tab stops and selected configuration parameters for ***each edit buffer***. For example, this lets you have word wrap enabled for one file being edited, but not for another.

The edit buffer dependent configuration parameters are identified with a "(*)" in their name. Included are many of the programming, word processing and file handling parameters.

Changing any of these parameters always affects the current edit buffer. It also sets the value that will be used for any newly opened buffers and the value that will be saved by **{CONFIG, Save config}**. (See Notes: below for exceptions.)

When "Config all buffers" is enabled, changing any of these parameters also affects all other edit buffers (files) that are currently open.

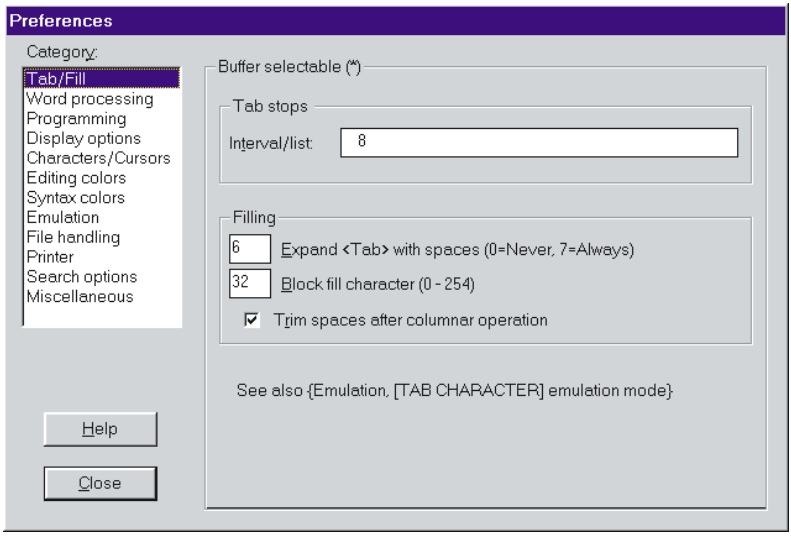
NOTE: This item is enabled by default because other editors do not have this level of flexibility and new users might otherwise be confused. It is very likely that you will want to disable it at times.

For example, if you are editing a program source code file and a documentation file, you may want to have **{CONFIG, Word processing, Word wrap}** enabled in the documentation file. However, since accidentally selecting paragraph formatting would scramble your program, you only want to enable "Word wrap" for the one file.

Notes:

Changing **{CONFIG, File handling, File type}** and **{CONFIG, File handling, Record header size}** only affect the current edit buffer and not other edit buffers, regardless of how **{CONFIG, All buffers}** is set. Also these parameters are not saved into the **`vedit.cfg`** file.

Tab and Filling



Tabs stops (*) [Default = Every 8 columns]

If only one number is entered, it is considered a uniform tab interval. Since the far left column is column number 1, the first tab stop will be at the interval plus 1. Therefore, the default value of "8" sets the tab stops at column 9, 17, 25, 33, 41, ...

If more than one number is entered, they are considered to be explicit tab stops.

If for some reason you only wanted a single tab stop, for example "20", enter it as "20 20".

Any Tab characters in your file past the last explicit tab stop are displayed as normal control characters.

You can set a separate set of tab stops for each edit buffer if you disable **{CONFIG, Config all buffers}**.

Expand <Tab> with spaces (*) [Default = 4]

Determines whether the **<Tab>** key (**[TAB CHARACTER]** function) inserts a Tab character or, alternatively, spaces to the next tab stop.

Also controls whether whitespace in a columnar block that is being inserted, copied or moved is re-tabbed to the optimal number of Tab characters. Alternatively, any Tabs are converted to just spaces. This switch does not affect normal non-columnar block operations.

Also controls whether a "Block fill" with spaces will actually insert the optimal number of Tab characters and spaces.

This option combines several options into one by having you add "mask" values (setting bits) for each desired sub-option.

The base value of "0" has VEDIT convert to Tab characters whenever possible. With a value of "7", VEDIT never inserts Tab characters.

Mask 1 The <Tab> key is expanded with spaces. Left margin indentation is with only spaces.

Mask 2 Tab characters within a columnar block operation are converted to spaces.

Mask 4 The spaces in a "Block fill" operation are never converted to Tabs.

Example values:

0 The <Tab> key is not expanded - a Tab character is entered into text. Whitespace in a columnar operation is re-tabbed. Spaces in a "Block fill" are converted to the optimal number of Tabs.

1 The <Tab> key is expanded with spaces. Whitespace in a columnar operation is re-tabbed. Spaces in a "Block fill" are converted to the optimal number of Tabs.

4 (Default) The <Tab> key is not expanded - a Tab character is entered into the text. Whitespace in a columnar operation is re-tabbed. Spaces in a "Block fill" are not converted to Tabs.

7 The <Tab> key is expanded with spaces. Whitespace in a columnar operation is not converted to Tabs. Spaces in a "Block fill" are not converted to Tabs.

Other than columnar block operations, this option does not affect Tab characters already in your file. Use **{BLOCK, Edit/translate, Detab}** to convert existing Tab characters in your file to spaces.

Block fill character (*) (0 - 255) [Default = 32]

This is the character used by **{BLOCK, Edit/Translate, Block fill}** and **{BLOCK, Edit/Translate, Insert empty column}** to fill/insert blocks. It is also used when selecting the **[ ] Fill buffer text** option with **{BLOCK, Move to register}**.

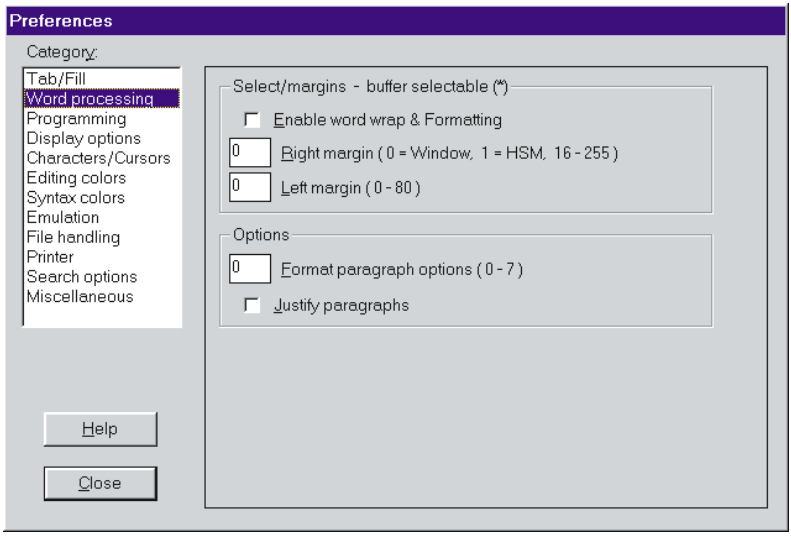
It is typically set to "Space" (value 32), but can be changed for special purposes. For example, setting it to a "." (value 46) lets you insert a column of periods with **{BLOCK, Edit/Translate, Insert empty column}.**

Note: Depending upon the setting of "Expand <Tab> with spaces", VEDIT may convert spaces to the optimal number of Tabs and spaces.

Trim spaces after columnar operation (*) [Default = Yes]

Controls whether the trailing spaces at the end of lines are removed following a columnar block operation. This only trims spaces on those lines involved in the operation.

Word Processing



Enable word wrap and formatting (*) [Default = No]

Enables word wrap when entering new text and paragraph formatting with **{EDIT, Format paragraph}**. It should be disabled when editing programs!

VEDIT only performs word wrap when entering text past the right margin. Use **{EDIT, Format paragraph}** to reformat existing text.

If **{CONFIG, Config all buffers}** is disabled, you can enable or disable word wrap for each file being edited.

Right margin (*) (0=Window, 1=HSM, 16 - 255) [Default = 0]

Sets the right margin used for word wrap, **{EDIT, Format paragraph}** and **{EDIT, Center line}**. The default value of "0" sets the right margin to the current window width. "1" sets it to the same value as **{CONFIG, Display options, Horizontal scroll margin}**. Any desired value between 16 and 255 can also be set.

A value of "70" is good for printing text on an 80 column printer with **{CONFIG, Printer, Left margin}** set to 10, which gives a 1 inch left margin.

(The right margin is actually set to one less than the width of the window or the horizontal scroll margin to leave room for the cursor.)

Left margin (*) (0 - 80) [Default = 0]

Sets the left margin used for formatting paragraphs. It can also be changed with **{EDIT, Indent}** (**<F8>**) and **{EDIT, Undent}** (**<F7>**).

Note: Don't set a left margin to keep your text from printing on the left edge of the paper. Instead use **{CONFIG, Printer, Left margin}** to position your text on the printed paper.

Format paragraph options (0 - 7) [Default = 0]

Determines how **{EDIT, Format paragraph}** deals with extra spaces.

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" trims trailing spaces, removes extra spaces from between words and leaves only a single space following ".", "!" and "?".

Mask 1 Add a trailing space after each paragraph line except the last. This extra space is allowed to exceed the right margin.

Mask 2 Allow extra spaces between words.

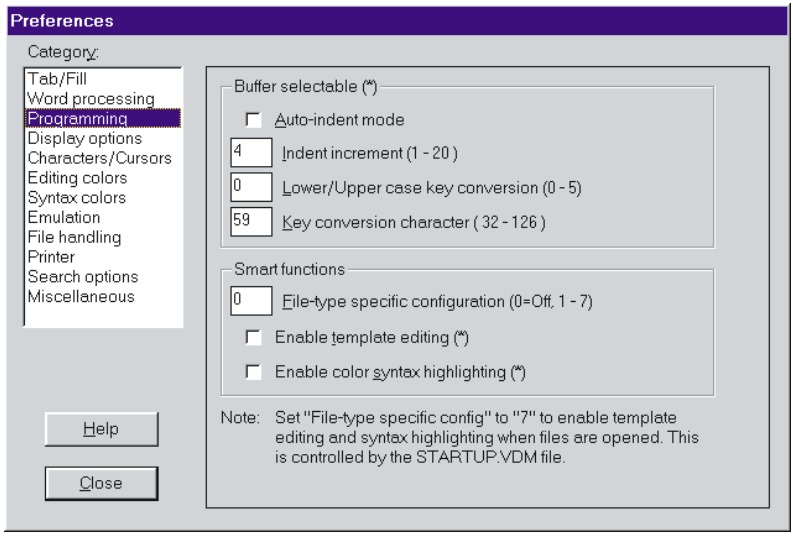
Mask 4 Allow two spaces after ".", "!" and "?". Use this if you like two spaces between sentences.

For example, if you want trailing spaces after each line and two spaces between sentences, configure this parameter to "5".

Justify paragraphs [Default = No]

When enabled, **{EDIT, Format paragraph}** will also justify the formatted paragraph by adding spaces between words to give an even right margin.

Programming



Auto-Indent mode (*) [Default = No]

When enabled, the indent position (left margin) for a new line of text is initially the same as for the previous line of text. This is convenient for programming in 'C', Pascal, etc. It is sometimes desirable when editing word processing documents.

The indent position can be changed with **{EDIT, Indent}** (**<F8>**) and **{EDIT, Undent}** (**<F7>**).

Indent increment (*) (1 - 20) [Default = 4]

Controls how much the left margin is indented/undented for each **{EDIT, Indent}** and **{EDIT, Undent}**.

Common values are "4" or the same value as the tab stop interval.

Lower/Upper case key conversion (*) (0 - 5) [Default = 0]

Determines whether lower case letters typed on the keyboard are converted (inserted) as upper case letters. Upper case can also be converted to lower case.

Primarily for assembly language programming. It does not affect any existing text; use **{EDIT, Edit/translate, Lower/Upper case}** to convert existing text.

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 "Key conversion character" (below). Primarily applicable to assembly language programming, where it is desirable to have 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.

4 All upper case letters are converted to lower case.

5 Similar to (2) except that upper case letters are conditionally converted to lower case.

Key conversion character (*) (32 - 126) [Default = ";"]

Sets the conditional lower/upper case key conversion character used by "Lower/upper case key conversion" (above). The default ";" is applicable to assembly language programming.

File-type specific config (0=Off, 1 - 7) [Default = 0]

Determines whether VEDIT configures itself according to common filename extensions such as ".txt", ".c", ".html", etc. The **`startup.vdm`** file determines the configuration performed for each file type, including color syntax highlighting and template editing. The supplied **`startup.vdm`** file defines the following values:

0 All file-type specific configuration is disabled.

1 Enabled, but syntax highlighting and template editing are disabled.

3 Template editing is also enabled, syntax highlighting is disabled.

5 Syntax highlighting is also enabled, template editing is disabled.

7 Syntax highlighting and template editing are both enabled.

NOTE: This function only works with the supplied **`startup.vdm`** file.

Technical:

When enabled, the special "file open" event macros in text registers 110 and 112 are automatically executed for each file that is opened. Also the "file close" event macros in text registers 111 and 113 are automatically executed for each file that is closed and saved to disk.

See also:

The topics "File-type Specific Configuration", "Template Editing" and "Syntax Highlighting" in Chapter 5 (Advanced Topics).

Enable template editing (*) [Default = No]

When enabled, the special command macro in text register 115 is executed for each normal text character entered in Visual Mode. This macro is typically setup by **`startup.vdm`** to perform template editing or shorthand expansion.

Selecting **{MISC, Load template file}** automatically enables this setting in the current buffer (file).

See also "Template Editing" in Chapter 5

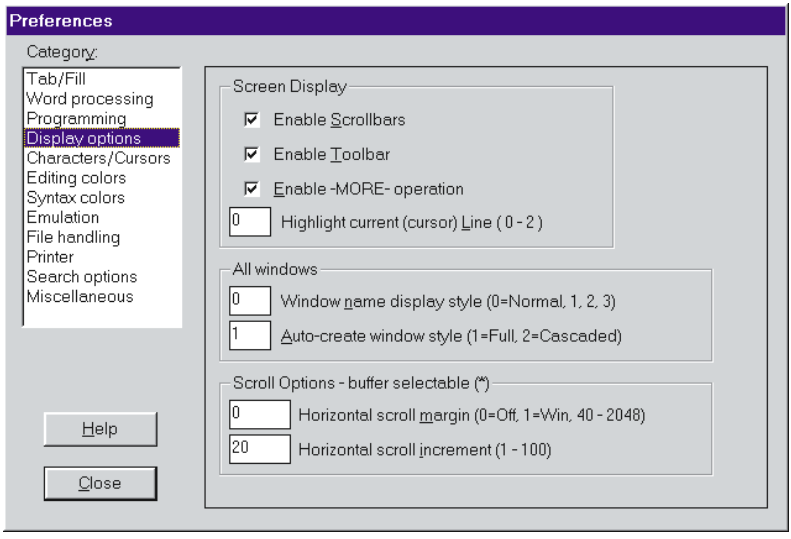
Enable color syntax highlighting (*) [Default = No]

When enabled, the text is color highlighted according to the currently loaded syntax definition file. (Note that screen updates will be slower; however this is barely noticeable on fast computers.)

Selecting **{MISC, Load syntax file}** automatically enables this setting in the current buffer (file).

See also "Syntax Highlighting" in Chapter 5. (Advanced Topics)

Display Options



Enable Scroll Bars [Default = Yes]

(Windows version only) When enabled, scroll bars are displayed for all editing windows. However, the vertical scroll bar is only displayed when the file (buffer) contains enough lines to require scrolling. Non-editing windows created with the VEDIT PLUS macro language never have scroll bars.

You can also turn scroll bars on/off with **{VIEW, Scroll bars}**.

Select **{CONFIG, Save config}** to make the configuration change permanent.

Enable Toolbar [Default = Yes]

(Windows version only) When enabled, the VEDIT toolbar is displayed. The toolbar is an easy way to access commonly used menu items with the mouse. A VEDIT program size of about 720 pixels is needed to see all toolbar buttons.

You can also turn Scroll bars on/off with **{VIEW, Toolbar}**.

Select **{CONFIG, Save config}** to make the configuration change permanent.

Enable -MORE- operation [Default = Yes]

When enabled, the screen display will be paused with the "-MORE- ..." prompt when any command macro attempts to display more than one "page" of text between keystrokes. When set to "No", text can scroll off the screen before it can be read.

Highlight cursor line (0 - 2) [Default = 0]

Determines whether the line that the cursor is on is highlighted. This can help you determine which line you are on.

0 The cursor line is not highlighted.

1 Full - The entire line (to the right edge) is highlighted.

2 Partial - Only text characters on the cursor line are highlighted. This makes it easy to see extra spaces past the ends of lines.

By default, the highlighted line is displayed using "bright" characters. Or select any desired color with **{CONFIG, Editing colors, Cursor line}**.

Window borders (1=Partial, 3=Scroll bars) [Default = 3]

(DOS, QNX, UNIX only) Determines the type of borders editing windows will have and whether scroll bars are displayed.

0 No borders. It is difficult to tell one window from another unless each window is in a different color.

1 Windows have minimal borders. When two or more windows are on the screen, each window has a top border. When necessary, windows also have a left-hand border.

2 Windows have full borders, but no scroll bars.

3 Windows have full borders with scroll bars. Scroll bars are only displayed in editing windows. Non-editing windows created with the VEDIT PLUS macro language will have full borders but no scroll bars.

Window name display style (0=Normal, 1 - 3) [Default = 0]

Determines how a window's name is displayed on its title bar.

0 Both the ID number and the buffer number may be displayed; however, in the usual case that they are the same, only the ID number is displayed.

1 Only the window ID number is displayed, e.g. "<2>".

2 Only the buffer number is displayed, e.g. "[1]" or "[1:2]".

3 Both the window ID number and the buffer number are always displayed.

Auto-create window style (1=Full, 2=Cascaded) [Default = 2]

Determines the size of windows that are auto-created for each additional file (buffer) that is opened.

Set to "1", a full-size overlapping window is created for each file. These windows can be resized/moved, cascaded or tiled with the **{WINDOW}** menu. **{VIEW, Reset}** initializes the screen and recreates one window for each file (buffer).

Set to "2", a smaller, cascaded window is created for each file.

The default is "Cascaded" because this is common with other editors and programs. However, we suggest "Full-sized" so that more of the file can be viewed without having to zoom all windows.

DOS, UNIX, QNX versions: A value of "0" allows a single window to be shared by all buffers. See the on-line help for this item.

Horizontal scroll margin (*) (0=Off, 1=Win, 40 - 2048] [Default = 0]

Determines whether lines longer than the window's width simply extend past the right edge (and are accessed via horizontal scrolling) or are wrapped onto multiple screen lines, called *continuation lines*.

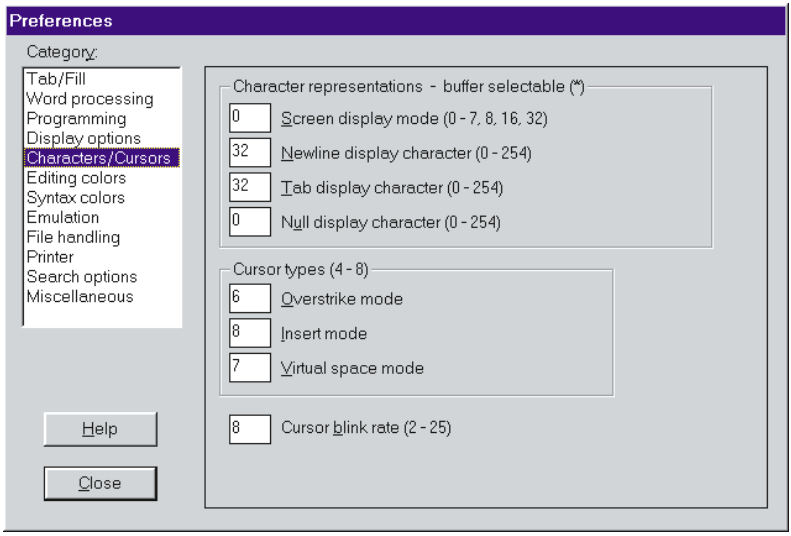
"1" sets the "scroll margin" to the window's width; lines wrap at the window's right edge. Other scroll margins between 40 and 2048 can also be set; lines wrap at the specified margin.

The default value of "0" disables screen wrapping; long lines extend indefinitely to the right.

Horizontal scroll increment (*) (1 - 100) [Default = 20]

Determines by how many columns the screen scrolls right or left when **[SCROLL RIGHT]** and **[SCROLL LEFT]** are pressed or VEDIT scrolls automatically.

Characters / Cursors



Screen display mode (*) (0 - 7, 8, 16, 32) [Default = 0]

Determines how ASCII control and graphics characters are displayed. It can also enable the Hexadecimal, Octal and EBCDIC modes for all characters.

Changing this value changes the display mode for the current window and all subsequently created windows; it does not affect other existing windows. The new display mode will be used for all windows if you select **{VIEW, Reset}**.

This option combines several options into one by having you add "mask" values (setting bits) for each desired sub-option.

The base value of "0" displays all characters literally, except <Tab>, <CR> and <LF>.

Mask 1 Display control characters in the format ^*x*.

Mask 2 Display graphics character in the format <*nnn*>.

Mask 4 Display <Tab>, <Null>, <CR> and <LF> literally. Used in the ASCII window following **{VIEW, Toggle Hex-mode split}**.

Mask 8 Display all characters in hexadecimal. Used in the hex-mode window following **{VIEW, Toggle Hex-mode split}**.

Mask 16 Display all characters in octal.

Mask 32 Display all characters in EBCDIC, or according to the currently loaded translate table. The file itself is not translated.

Mask-8, Mask-16 and Mask-32 cannot be combined with any other masks. Therefore, the valid values are "0" through "7", "8", "16" and "32".

{VIEW, Toggle display mode} toggles through the eight most useful values.

Normal ASCII files display as gibberish in EBCDIC mode.

Newline display character (*) (0 - 254) [Default = 32]

Determines the character displayed at the end of each line where the "newline" (<CR><LF> pair) is. The default "space" (value 32) is, of course, not visible. Visible candidates on an IBM PC include values "14", "17" and "20".

Tab display character (*) (0 - 254) [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 "250".

NOTES: This character is only used for screen display; spaces are always used when printing.

Use **{BLOCK, Edit/translate, Detab}** to convert tab characters in the file to spaces.

Null display character (*) (0 - 254) [Default = 0]

The IBM PC displays the "Null" character (value 00) as a space which is therefore indistinguishable from a real space (value 32). To make the Null character stand out better, you can display it as any other character. Values "4" and "7" are reasonable choices.

NOTE: This character is also used when printing because most printers ignore null characters. This makes it possible to see null characters in a printout.

Cursor type in overstrike mode (0 - 6) [Default = 2]

Cursor type in insert mode (0 - 6) [Default = 1]

Cursor type in virtual space mode (0 - 6) [Default = 0]

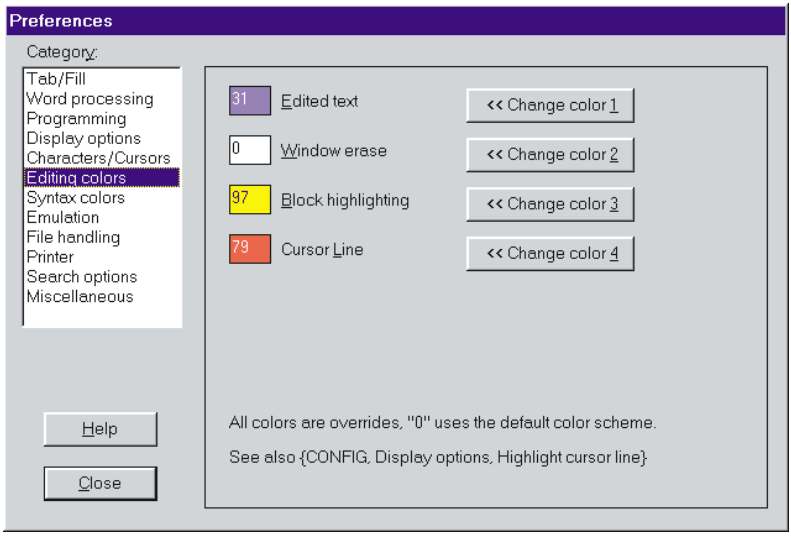
Determines how the cursor is displayed when the editor is in "Overstrike" or "Insert" mode, or in "Virtual space mode" — the space past the last character of a line.

Since the Windows, DOS, QNX and UNIX versions are all different, please refer to the on-line help for this item for the actual values.

Cursor blink rate (2 - 25) [Default = 8]

Determines the cursor's blink rate for the blinking cursor types above. A smaller number causes the cursor to blink faster. (The unit is 1/18 of a second.)

Editing Colors



Windows Version:

The configurable colors are split into the two categories “Editing colors” and “Syntax colors”; the latter are used only when Syntax highlighting is enabled.

All “Editing colors” can be set to “0” to use a default color, some of which are set by the overall Windows color scheme. Alternatively, explicit “overriding” colors can be selected.

DOS, UNIX, QNX Versions:

All colors used by VEDIT can be configured, including the colors in the main menu, the sub-menus and window borders.

{CONFIG, Colors} contains sub-menus to separate the many colors into logical groups. A color chart is displayed when you enter these sub-menus; however the DOS color chart does not support “point and shoot” color selection.

VEDIT has two sets of screen attributes; one for Color Displays and one for Monochrome displays. **{CONFIG, Colors, Enable monochrome}** selects which set is being used and configured.

The invocation option “**-m**” forces this setting to monochrome.

The “monochrome” attributes can be used as an alternate set of colors. For example, you may prefer one set on your desktop computer and another set on your laptop. A single **`vedit.cfg`** file could be configured for both computers. Use the “**-m**” invocation option when running on the Laptop. (The “VEDIT” environment variable can be used to make “**-m**” the default.)

Only the colors used in the Windows version are listed here. Refer to Chapter 8 (Configuration) or the on-line help in the **{CONFIG, Colors}** sub-menus for a description of parameters not covered here.

Edited text

Sets the primary color for the text in windows. The color of individual windows can subsequently be changed with **{MISC, Color Toggle}** (**<Alt-J>**). The initial value is set during installation.

Windows version: A value of "0" uses the default Windows color scheme. However, we recommend setting an explicit value so that it can be coordinated with the syntax highlighting colors. For example, white-on-blue (value 31) or yellow-on-blue (value 30) are good editing colors.

DOS version: An explicit value must be set; do not use "0" because that results in invisible text.

Window erase

Sets the color for those portions of a window where there is no text. The normal setting of "0" uses the same color as for "Edited text". Setting an overriding color gives an unusual effect, but lets you clearly see trailing spaces at the ends of lines.

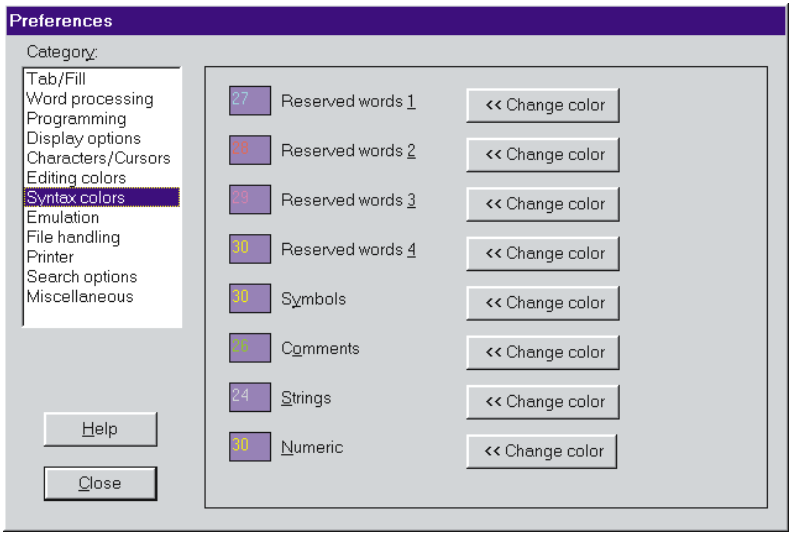
Block highlighting

With the default value of "0", highlighted blocks are displayed in reverse video. Use this parameter to set an overriding color. For example, "71" displays blocks as white text on a red background.

Cursor line

This parameter sets the color for the entire cursor (current) line when **{CONFIG, Display options, Highlight cursor line}** is enabled. The default setting of "0" causes the line to be displayed with the IBM PC "intensity" flipped.

Syntax Colors



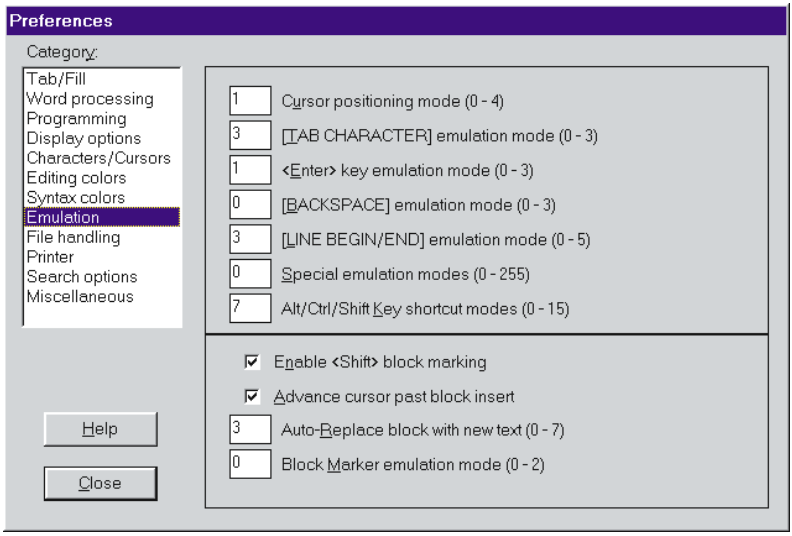
The syntax highlighting colors are only used when **{CONFIG, Programming, Enable color syntax highlighting}** is set.

The syntax highlighting definition file sets up pattern matching to recognize different parts of the displayed text as “Reserved words”, “Symbols”, “Comments”, “Strings” and “Numeric”. The matched text is then displayed in the corresponding color.

These colors must be explicitly set; there is no default value. ***DO NOT*** use a value of “0”, because that results in invisible text.

For the most pleasing visual effect, these colors should be coordinated with the color for “Edited text”. For example, if the edited text is white-on-blue, all of the syntax highlighting colors should also use a blue background.

Emulation



Cursor positioning mode (0 - 4) [Default = 1]

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 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. Use this mode for 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.

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

{CONFIG, Emulation, Special emulation modes} can be set to prevent **[CURSOR RIGHT]** and **[CURSOR LEFT]** from wrapping to the next/previous line when they reach the end/beginning of the line.

[TAB CHARACTER] emulation mode (0 - 3) [Default = 3]

Controls how the **[TAB CHARACTER]** and **[BACKTAB]** functions operate. These functions are almost always assigned to <**Tab**> and <**Shft-Tab**>.

0 **[TAB CHARACTER]** always inserts a Tab character (or spaces) to the next tab stop.

1 **[TAB CHARACTER]** performs **[NEXT TAB STOP]** in "Overstrike" mode and inserts a Tab character (or spaces) in "Insert" mode.

2 If the cursor is not in a highlighted block, **[TAB CHARACTER]** inserts a Tab character (or spaces). If the cursor is in a highlighted block, **[TAB CHARACTER]** and **[BACKTAB]** are equivalent to **{EDIT, Indent}** and **{EDIT, Undent}** — all lines in the block are re-indented.

3 Combines (1) and (2).

<Enter> key emulation mode (0 - 3) [Default = 1]

Controls how the <**Enter**> key (**[RETURN]** function) operates.

0 <**Enter**> always inserts a "newline" at the cursor position; this splits the current line or adds a new one at the cursor position.

1 <**Enter**> only inserts a "newline" in "Insert" mode; in "Overstrike" mode it is equivalent to **[NEXT LINE]** — it only moves the cursor to the beginning of the next line.

2 <**Enter**> is always equivalent to **[NEXT LINE]**; it never inserts a "newline".

3 <**Enter**> inserts a "newline" following the current line (it opens a new line).

[BACKSPACE] emulation mode (0 - 3) [Default = 0]

Sets the behavior of the **[BACKSPACE]** key.

0 **[BACKSPACE]** always deletes the preceding character. Pressed at the beginning of a line, it deletes the preceding "newline".

1 **[BACKSPACE]** only deletes in "Insert" mode. In "Overstrike" mode it only moves the cursor left to the previous character.

2 **[BACKSPACE]** always deletes the preceding character, but stops at the beginning of the line.

3 **[BACKSPACE]** only deletes in "Insert" mode and stops at the beginning of line.

[LINE BEGIN/END] emulation mode (0 - 5) [Default = 3]

Sets the behavior of **[LINE BEGIN]** and **[LINE END]** (normally the **<Home>** and **<End>** keys).

0 **[LINE BEGIN]** and **[LINE END]** move the cursor only to the first/last character currently displayed in the window. Since the window is not horizontally scrolled, this may not be the first/last character of the text line. Successively pressing **[LINE BEGIN]** or **[LINE END]** has no effect. (This mode is only useful with slow CRT terminals under UNIX or QNX.)

1 **[LINE BEGIN]** and **[LINE END]** move the cursor to the very first/last character of the current text line. The window is horizontally scrolled if necessary. Successive presses have no effect.

2 Move the cursor only to the first/last character displayed in the current window without scrolling. However, pressing **[LINE BEGIN]** and **[LINE END]** repeatedly moves the cursor to the preceding/next window line.

3 Move the cursor to the very first/last character of the text line. Pressing **[LINE BEGIN]** and **[LINE END]** repeatedly moves the cursor to the preceding/next screen line.

4 Move the cursor only to the first/last character displayed in the current window. Otherwise same as "5". (Probably of limited use.)

5 Move the cursor to the very first/last character of the text line. Pressing **[LINE BEGIN]** a second time moves to the top of the screen. Pressing **[LINE BEGIN]** a third time moves to the beginning of the file. Similarly, pressing **[LINE END]** a second time moves to the bottom of the screen. Pressing **[LINE END]** a third time moves to the end of the file. (This emulates the Brief (tm) editor.)

Special emulation modes (0 - 255) [Default = 0]

Enables special emulation modes. "0" disables the special modes.

This option combines six options into one by having you add "mask" values (setting bits) for each desired sub-option.

Mask 1 **[SCREEN BEGIN]** and **[SCREEN END]** go to the first/last column instead of remaining in the current column.

Mask 2 **[SCROLL UP]** and **[SCROLL DOWN]** leave the cursor in the current screen line instead of in the current text line.

Mask 4 **{SEARCH, Search}** and **{SEARCH, Replace}** also perform **{SEARCH, Next}**. You must first press **[CANCEL]** and then **{SEARCH, Search}** to enter a new search string.

Mask 8 **[CURSOR RIGHT]** and **[CURSOR LEFT]** don't wrap to the next/previous line when they reach the end/beginning of the line.

Mask 16 The last normal character, not the "newline" is the last accessible character on the line. (Used by the "vi" emulation).

Mask 32 **[DELETE]** doesn't delete the "newline".

Mask 64 **[DELETE]** doesn't delete a highlighted block. Otherwise, it will delete a highlighted block when both block markers are set and the cursor is within the block. Blocks can always be deleted with **{BLOCK, Edit/translate, Block delete}**.

Mask 128 **[DELETE]** doesn't delete a highlighted block if only 1 marker set; both markers must be set.

Alt/Ctrl/Shift key shortcut modes (0 - 15) [Default = 7]

Determines if tapping the <**Alt**>, <**Ctrl**> and <**Shift**> keys perform shortcut functions. This option combines six options into one by having you add "mask" values (setting bits) for each desired sub-option.

The base value of "0" disables all shortcuts.

Mask 1 Tapping <**Alt**> performs **[MENU]**; tapping it again removes the menus. Note: this is always enabled in the Windows version.

Mask 2 Double-tapping <**Ctrl**> performs **[CANCEL]**; this is an easy way to remove block markers.

Mask 4 Double-tapping <**Shift**> performs **{BLOCK, Set stream marker}**; this is an easy way to set block markers.

Mask 8 <**Ctrl+Shift**> performs **[ERASE LINE]**; this is an easy way to delete lines. (This is not the default because it requires some care; if you need both keys down for "<Shift> block marking", press <**Shift**> first.)

Enable <Shift> block marking [Default = Yes]

Controls whether the <**Shift**> key can be used to mark blocks of text.

When enabled, a block can be marked (highlighted) by holding down the <**Shift**> key and moving the cursor. This includes the "arrow"" keys, <**Home**>, <**End**>, <**PgUp**>, <**PgDn**>, <**Ctrl-End**>, etc. In this mode, <**Shift-Home**>, etc., cannot be used as function keys. Also, <**Shift-F1**> through <**Shift-F12**> cannot be used as cursor movement functions, but can be used for other functions.

When disabled, <**Shift-Home**>, etc., can be used as function keys with editing operations and have keystroke macros assigned to them.

Auto-replace block with new text (0 - 7) [Default = 3]

Determines whether a highlighted block of text at the cursor position is automatically deleted if new text is typed or inserted from a text register or the clipboard. This is the default and is typical for most Windows programs. See NOTES below.

For example, a search normally highlights the found text as a block. You can then immediately type in replacement text without first deleting the block.

Hint: To insert new text without deleting the searched text, first press **<Cursor Right>** and then **<Cursor Left>** to remove the block highlighting. Or double-press **<Ctrl>**.

0 Off. A highlighted block is never automatically deleted.

1 Enabled only for new typed text. Largest block size is 1K bytes.

2 Enabled only for inserting text registers or the clipboard.

3 Enabled for both new text and inserting a register or clipboard.

7 Enabled for both. Auto-delete blocks of any size.

NOTES: As a safety feature, the largest block that can be auto- deleted is normally 1000 bytes. For larger blocks, simply press **<Del>** first.

The value "7" bypasses the 1000 byte limit. Use with care!

We suggest turning this feature off if you edit files with critical data.

Advance cursor past block insert [Default = Yes]

Determines whether the cursor advances following a block, scratchpad, text register or clipboard insertion. It also determines the cursor position following **{EDIT, Insert file}**.

No The cursor is not moved, and is left at the beginning of the inserted block. Useful when you want to edit at the beginning of the inserted block.

Yes The cursor is advanced past the inserted text.

Block marker emulation mode (0 - 2) [Default = 0]

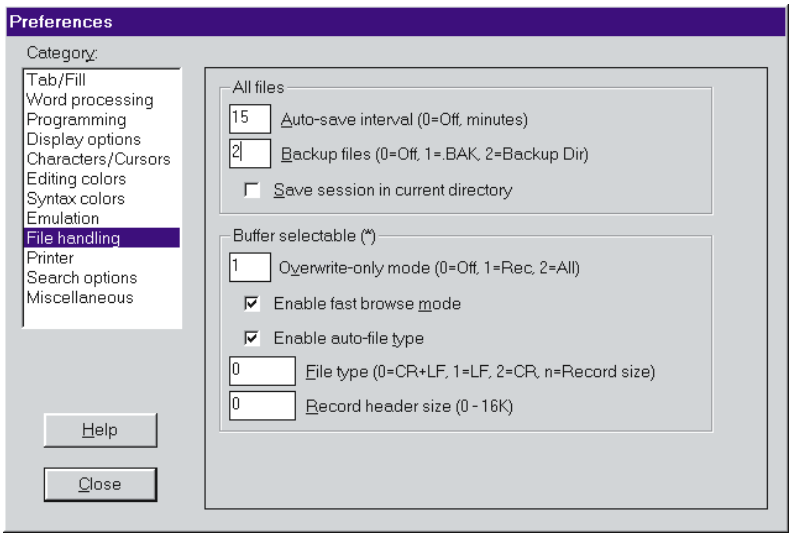
Determines whether the new block of text is highlighted following a **{BLOCK, Copy to cursor}** or **{BLOCK, Move to cursor}**.

0 The block markers are automatically removed (cleared) following a block copy or move.

1 With a Copy, the original block remains highlighted; with a Move, the block is highlighted in its new position. This emulates real WordStar. It simplifies copying a block to several positions, but requires an extra step to remove markers.

2 With both a Copy or Move, the block is highlighted in its new position. This emulates the Borland editors (emulation of WordStar).

File Handling



Auto-save interval [Default = 0]

VEDIT can optionally auto-save all modified files after a configurable number of minutes. This option sets the time interval. A value of "0" disables this feature. Although by default this feature is turned off, we recommend that you enable it with a typical value of "20".

Backup files (0=Off, 1=.BAK, 2=Backup Dir) [Default = 1]

Controls whether backup files are created so that you can refer back to the original file if needed. We ***HIGHLY*** recommend that backup files be enabled.

0 Disabled. Backup files are not created.

1 Create backups by renaming the original file to have a ".BAK" filename extension. (UNIX, QNX: ".b" extension.)

2 Create backups by moving the original file to the *VEDIT Backup Directory*, typically "C:\VEDIT\BACKUP" or "C:\BACKUP".

See also "Exiting VEDIT - Backup Files" in Chapter 4.

Save session in current directory

When enabled, the edit session save/restore files **`veditsav.env`** and **`veditsav.dat`** are saved in the current directory. This permits restoring the last edit session you had in that directory.

Otherwise, **`veditsav.env`** and **`veditsav.dat`** are saved in the *User Config Directory*, typically "C:\VEDIT". When VEDIT is next invoked (without filenames) from anywhere, you will be switched to the last directory you were in and the files you were last editing.

Use V-SWAP when entering DOS [Default = Yes]

DOS version only: Determines whether VEDIT will use the V-SWAP program (when already installed in memory) to swap itself out of memory when entering DOS. No error is given if V-SWAP is not in memory.

Overwrite-only mode (*) [Default = 1]

Controls whether the current file is in overwrite-only mode. In this mode deletions and insertions which would change the size of the file cannot be made; however character overstriking and block overwrites can be made.

0 Disabled. All editing changes can be made to any type of file. (DOS VEDIT PLUS: Disk sector editing is always in overwrite mode).

1 Record mode. Overwrite-only mode is enabled if the "File type" is set to "8" or greater for fixed-length-record data or binary files.

2 Enabled for all file types.

Enable fast browse mode (*) [Default = Yes]

When VEDIT opens a file in "Read-only" mode, it does not always know the current line number; the line number is then displayed as "?????". For example if you have opened a multi-megabyte file and select **{GOTO, End of file}**, VEDIT will instantly jump to the end of the file without counting the number of lines in the file.

When disabled, browsing will be slower, but the correct line number will always be displayed.

Enable auto-file type (*) [Default = Yes]

When enabled, VEDIT examines each newly opened file to determine its most likely file type, i.e. as a "Window/DOS text", "UNIX text", "Mac text" or "binary/data" file.

When disabled, VEDIT does not examine the file, and the file type is simply set by **{CONFIG, File handling, File type}**.

Since you can easily override the file type, there is little reason to disable this feature.

File type (*) (0=CR+LF, 1=LF, 2=CR, n=record size)

Determines the type of file VEDIT assumes it is editing. It controls the screen display, what the "newline" character is, and changes the behavior of some editing operations.

VEDIT automatically determines the "most likely" file type of each opened file if **{CONFIG, File handling, Enable auto-file type}** is enabled. You can override this value if VEDIT sets it incorrectly for your needs.

0 Lines end in Carriage-Return and Line-Feed. Typical for Windows/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 UNIX/QNX. 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.

2 Lines end in just a Carriage-Return. Typical for Mac. Line-Feed is no longer a special character and if it occurs in the text it is displayed as "<LF>". The **<Enter>** key inserts only a Carriage-Return.

n Values of "8" through "2048" correspond to the record length for fixed-length-record data files. Values of "64" and "16" are useful for editing binary files. Instead of assuming that lines end in a "newline" character, each line is treated as simply 'n' characters. This is described in more detail under "File Types - Win/DOS, Unix, Mac, Binary" in Chapter 4.

By default, "Overwrite-only" mode is selected; change it if you need to delete or insert characters. Word processing functions are not available in this mode.

Notes:

Windows version: You can set the correct file type or record length in the File-open dialog box when you open the file.

You can temporarily toggle the file type to Binary-16 or Binary-64 with **{VIEW, Toggle binary/text mode}**.

Changing this value only affects the current edit buffer even if **{CONFIG, Config all buffers}** is enabled.

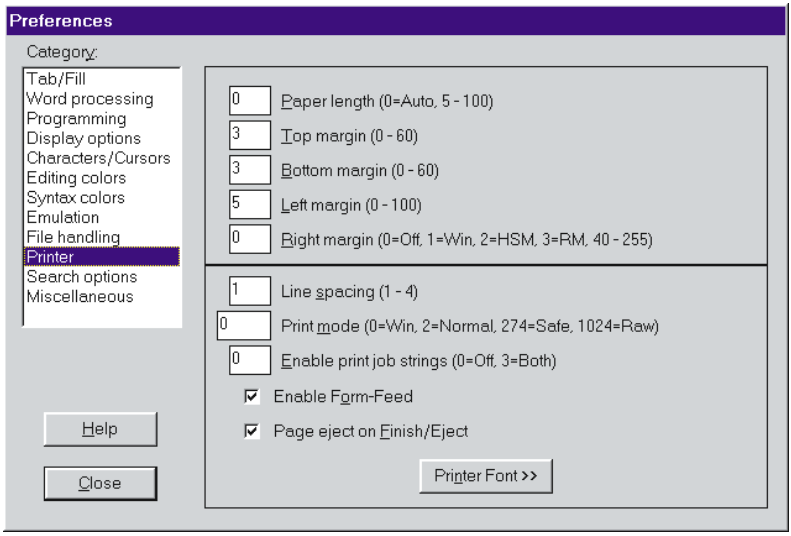
Record header size (*) (0 - 16K)

Some fixed-length-record data files begin with a header that is not the same length as the records. By setting this value to the length of the header and the file type to the length of the records, all records will be properly aligned within VEDIT. xBASE .DBF files are an example. This is described in more detail under "File Types - Data (Database) Files with Headers" in Chapter 4.

When set, the header will be "Line 0" and the line number will display the current record number.

NOTE: Changing this value only affects the current edit buffer even if **{CONFIG, Config all buffers}** is enabled.

Printer



Paper length (0=Auto, 5 - 100)

When set to "0=Auto" (Windows version only), the number of lines per page is automatically determined by the size of the current printer font, the printing orientation (portrait or landscape) and the paper size.

Otherwise, this value should be set to the length of the paper in lines. This is typically 66 for dot-matrix printers and between 58 and 62 for laser printers.

If this value is set too large, every other printed page will have just a few lines on it. If in doubt, set it to 58 or 60. You may want to change it when printing labels or short forms.

The number of lines of text printed per page is equal to the "Page length" minus "Top margin" minus "Bottom margin".

Notes:

Many laser printers have a default setting of 60 lines per page. This will work properly with a "Paper length" of 66 as long as **{PRINT, Config, Form-Feed}** is enabled and the top/bottom margins are each set to 3 or more.

Top margin (0 - 60) [Default = 3]

This value determines how many lines are left blank at the top of each page. A value of "0" causes text to print on the very first line.

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 = 5]

This value is set to avoid printing at the very left edge of the paper. Setting this value has no effect on editing; it has the same effect as adjusting the paper sideways in the printer.

With laser printers, a value of "5" typically gives a one inch left margin because the printer adds its own 1/4" - 1/2" margin. A value of "10" typically gives a one inch left margin on a dot-matrix 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.

Right margin (0=Off, 1=Win, 2=HSM, 3=RM, 40-255) [Default = 0]

Lines longer than this optional margin are wrapped onto multiple printed lines. This is useful when printing long lines because most printers otherwise truncate long lines. Values are:

0 Right margin feature is turned off.

1 Right margin is automatically set according to the current window width.

2 Right margin is set to the same value as **{CONFIG, Display options, Horiz. scroll margin}**.

3 Right margin is set to the same value as **{CONFIG, Word processing, Right margin}**.

40 - 255 Usually set to the number of characters your printer can print per line, typically 80, 96 or 132.

Line spacing (1 - 4) [Default = 1]

Determines whether the text is printed single spaced (1), double spaced (2) or triple spaced (3). For example, a value of "2" leaves a blank line between every printed line.

Print mode (0=Win, 2=Normal, 274=Safe, 1024=Raw)

Determines if control and graphic characters are sent "as-is" to the printer, or are first expanded. Also permits printing in hexadecimal, or printing an EBCDIC file on an ASCII printer. The most common values are:

0 Characters are printed in the same mode as they are displayed; the display mode is changed with **{VIEW, toggle display mode}** or **{CONFIG, Characters/Cursors, Screen display mode}**.

2 Tabs are expanded to spaces, all other control/graphics characters are sent as-is to the printer.

274 "Safe" mode. All control character are converted to the ^***x*** format so that they don't control the printer. It is a combination of Masks 256, 16 and 2.

1024 All control characters including Tabs are sent as-is to the printer.

Other print modes are selected by adding "mask" values (setting bits) for each desired characteristic:

Mask 8192 Print EBCDIC in equivalent ASCII, using translate table. Other translate tables can be loaded.

Mask 4096 Print all chars in octal.

Mask 2048 Print all chars in hexadecimal.

Mask 1024 Print all control characters, including Tabs and <Esc> as-is. (Overrides any Masks below.)

Mask 512 Print graphics characters in the format **<nnn>**.

Mask 256 Print control characters in the format **^*x***.

Mask 32 Print Carriage-Return and Line-Feed as **<CR>** and **<LF>**.

Mask 16 Print Backspace as **^H** (depends upon Mask 256).

Mask 8 Print Escape character as **<Esc>**.

Mask 2 Expand Tabs with spaces.

Mask 1 Assume tab stops at every 8 columns.

NOTE: When printing in hexadecimal or EBCDIC, you may need to set the "Right margin" so that long lines are wrapped onto multiple printed lines.

Enable print job strings (0=Off, 3=Both) [Default = 0]

Determines whether the "Print job start string" and "Print job finish string" are sent. The start string can be used to select a font; the finish string to reset the printer to its default font. Values are:

0 Don't send either string.

1 Send only the print job start (init) string.

2 Send only the print job finish (reset) string.

3 Send both strings.

The strings are set by editing the **`vedit.cfg`** file.

Windows Version: These strings are of limited use because Windows uses its own strings according to the type of printer. The "Print job finish string" is probably never needed.

DOS Version: You can override the default setting in the Printing dialog box. The strings can also be set in **{CONFIG, Printer}**.

Enable Form-Feed [Default = Yes]

When enabled, a single "Form-Feed" character is used to advance to the top of a new page; nearly all printers can be advanced this way. Otherwise, it uses the correct number of "Line-Feeds" to advance to the next page. A Form-Feed character in the file also signals VEDIT to start a new page.

When possible, it is better to enable this parameter; the "Paper length" value is then not so critical.

Paper orientation [Default = 0] (Windows only)

Printing is normally in portrait or landscape mode, depending upon the default settings for the printer. Alternatively, this parameter can force the printer into portrait or landscape mode. A value of “1" forces the printer into Portrait mode while ”2" forces the printer into Landscape mode.

Page eject on Finish/Eject [Default = Yes] (DOS only)

Determines whether a page eject is included at the end of each print-job, i.e. with printing dialog box items **( ) All** and **[Finish/Eject]**.

However, on some network printers and under UNIX/QNX, you may need to set this parameter to “No” to prevent blank pages after each print job.

Printer (0=Default ... 7=file) [Default = 0] (DOS only)

This parameter determines to which parallel or serial port VEDIT prints. You can also print to a file.

When set to “0”, it prints to the default printer as set by **{PRINT, Config, Change default printer}**, typically “PRN”. “PRN” is normally the same as “LPT1”; however, you can use the DOS ”MODE” command to reroute it to another parallel or serial port.

You can also print directly to a parallel or serial port, or a file:

1 LPT1
2 LPT2
3 LPT3
4 COM1
5 COM2
6 COM3
7 file

When set to “7”, VEDIT prints to a file. For each print job, it will prompt you for the name of the file. You can also print to the same file each time by setting this parameter to “0” and changing the “default” print device to be a filename.

Change default printer [Default = “PRN”] (DOS, UNIX, QNX only)

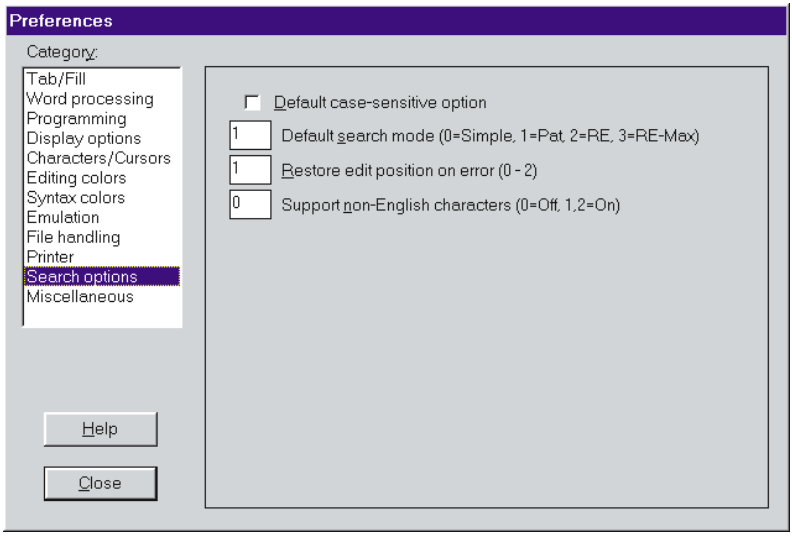
Selects the “default” device to which VEDIT prints. Under DOS, this is initially “PRN”. This device can also be changed to a filename such as “veditprn.prn”, However, each new print-job will then overwrite the previous one, unless you change the filename.

Under UNIX/QNX, the default print device is “lp”. It can be changed as necessary.

Printer Font

The text can be printed in any desired font and size. The editing (display) font and the printing font can be completely different. Unlike WYSIWYG Word Processors, with VEDIT you can edit in an easy-to-read screen font and print the text in a different font.

Search Options



Default case-sensitive option [Default = No]

When enabled, the search option **[ ] Case** is selected by default. This makes the search case sensitive. Otherwise the search is not case sensitive, e.g. the search string "this" will also locate "This" and "THIS".

If you want **{SEARCH, Incremental search}** or **{SEARCH, Compare buffers}** to be case sensitive, you must set this parameter.

Default search mode (0=Simple,1=Pat,2=RE,3=RE-Max) [Default = 1]

Sets the default search mode for the Search and Replace dialog boxes. Users not familiar with regular expressions should leave this set to "1". With regular expressions, many normal characters have a special meaning and cannot be searched "as-is".

0 Simple. No pattern matching or regular expressions are used.

1 (Default) Pattern matching is used. All pattern matching codes begin with "|"; enter "||" to search for a single "|" in the file. (Note: "|" is the "pipe" character, which is **<Shift>**-\ on the keyboard.)

2 Regular expressions are used. Most punctuation characters have a special meaning. Closely follows the UNIX standards.

3 "Maximized" regular expressions are used. Consider the regular expression search string "a.+b" and the text "12a3456b7890b". When maximized, it will match "a3456b7890b"; otherwise it will only match "a3456b".

See Also:

"Search and Replace" in Chapter 4.
{SEARCH, Search}, **{SEARCH, Replace}**

Restore edit position on error (0 - 2) [Default = 1]

Determines the position of the cursor following an unsuccessful search.

0 Positions the cursor as close to the pre-search position as possible, but without performing any file buffering.

1 Restores the cursor to the pre-search position.

2 Positions the cursor at the End-of-file. (Beginning-of-file for reverse searches).

Normally "1" is nice, but in huge files, "0" or "2" should be used to save time — it takes a noticeable amount of time to restore the edit position in multi-megabyte files.

Support non-english characters (0=Off, 1=ANSI, 2=On)

Determines whether the pattern matching codes "|**A**", "|**U**" and "|**V**" will match non-english letters in the extended character set with decimal value 128 - 255.

It also determines whether **{BLOCK, Edit/translate, Upper/Lower/Switch case}** recognizes non-english letters.

0 Off. None of the extended characters (128-255) are recognized as letters. This works best with English.

1 When an ANSI font is displayed, non-english letters in the extended ANSI character set are recognized. When an OEM font is displayed, non-english letters are not recognized.

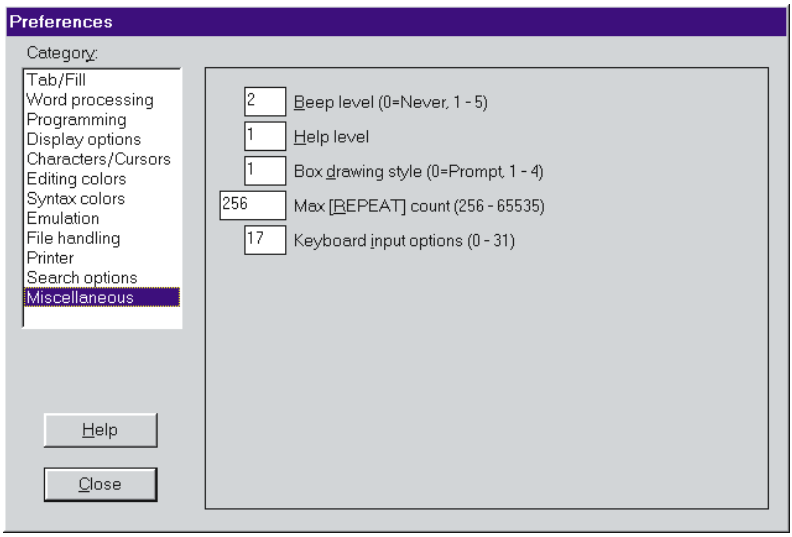
2 Non-english letters are recognized, depending upon the font. When an ANSI font is displayed, non-english letters in the extended ANSI character set are recognized. For example, decimal value 252 which is an umlaut "u" is treated as a lower case letter.

When an OEM font is displayed, non-english letters in the extended IBM-PC (OEM) character set are recognized. For example, decimal value 129 which is an umlaut "u" is treated as a lower case letter.

Note:

(Technical) VEDIT queries Windows for information about ANSI non-english letters. Therefore, if you notice any inconsistencies, be sure that Windows has been set to the correct language (code page).

Misc



NOTE: The DOS, UNIX and QNX versions contain additional items in this sub-menu. Refer to the on-line help for this sub-menu for a complete description.

Beep level (0 - 5) [Default = 2]

Controls under what conditions VEDIT produces a beep on the speaker.

0 VEDIT never beeps the speaker.

1 Only beeps under control of the macro language **Alert( )** command.

2 Also beeps for error messages and when answering prompts with invalid responses.

3 Also beeps when pressing invalid keys in the menus.

4 Also beeps when pressing unassigned control and function keys.

5 Also beeps when attempting to scroll past beginning/end of the buffer, or attempting to undo/redo when there is nothing more.

Help level (0 - 1) [Default = 1]

When set to “1”, a help line is displayed on the bottom of the screen whenever you are in the menu system, during point and shoot file selection and for selected other prompts. “0” turns this help line off.

Box drawing style (1 - 4) [Default = 1]

Determines the style used by **{MISC, Box drawing mode}** for drawing the vertical and horizontal lines with **<Shift+Cursor>**. Note that **<Ctrl+Cursor>** draws a different, complementary, type of line.

Maximum [REPEAT] count (1 - 65535) [Default = 256]

Determines the maximum repeat count that can be entered for **[REPEAT]** and **{EDIT, Repeat}**.

The default maximum of "256" is used to prevent novice users from inadvertently entering a huge count for an edit operation that might corrupt their file or take a long time to perform. Values larger than 256 are often desirable, but can exceed VEDIT's ability to undo them. Experienced users may prefer a larger value.

Keyboard input options (0 - 31) [Default = 17]

Controls processing of keyboard characters. You will only need to change this value for special applications.

This option combines five 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 enabled on an IBM PC.

Mask 2 — Treat 8 bit (graphics) characters as function keys. This is useful on some CRT terminals, but should be disabled on an IBM PC.

Mask 4 — Enter unassigned function/control keys into the text. Almost everyone will want this disabled.

Mask 8 — Reverse the case of all letters, e.g. typing "a" gives you "A" and typing "A" gives you "a". (We are not exactly sure why you would want this, but a few users have asked for it.)

Mask 16 — In a multi-character control sequence, convert the 2nd and following <Ctrl> characters to equivalent letters. This is useful for the WordStar layout so that, for example, ^K ^V is equivalent to ^K V. This is enabled by default because it does not interfere with other layouts.

The recommended values are "1" and "17".

Keyboard Layout (Sub-menu)

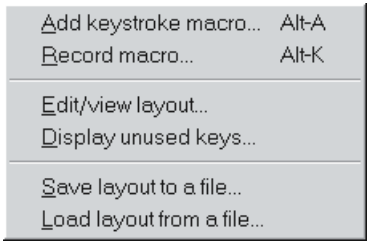
Keystroke Equivalent: (None)

Introduction:

VEDIT's keyboard layout is completely configurable. Each basic edit function can be assigned to any key or alternate keys. As many keystroke macros and "hot-keys" as desired can be added to the layout.

This sub-menu lets you record or add new keystroke macros and view/edit the entire keyboard layout as a normal text file. While editing, you can also print the layout if desired.

To make changes to the keyboard layout permanent, they must be saved to the **`vedit.key`** file. Entire keyboard layouts can also be loaded "on the fly".



Notes:

Experienced VEDIT users will probably prefer to make changes to the keyboard layout and add new keystroke macros by directly editing the **`vedit.key`** file. This is described in Chapter 8 (Configuration).

Keystroke macros and changing the keyboard layout are extensively covered in Chapter 4.

VEDIT's normal keyboard layout includes many pre-defined keystroke macros for selecting menu items; these are often called "hot-keys". VEDIT's menus display all assigned "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 **<F12>** as the "lead-in" to an entire group of keystroke macros. Pressing **<F12>** and then **"A"** could play back one keystroke macro, pressing **<F12>** 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 that 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 keyboard layout 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) that perform a single edit function are considered part of the basic keyboard layout as displayed by **{HELP, Keyboard layout}**; everything else is considered a keystroke macro.

Add Keystroke Macro

Define a new keystroke macro or change an existing one.

Keystroke Equivalent:

<Alt-A>, This is a keystroke macro.

Full Description:

This item adds a new keystroke macro or changes an existing one. Compared to **{CONFIG, Keyboard layout, Record keystroke macro}**, this item has the advantage of letting you edit the macro as you enter it. It should also be used for macros that access the VEDIT PLUS macro language.

➢ **To add (define) a new keystroke macro:**

1. Select **{CONFIG, Keyboard layout, Add keystroke macro}**.
2. At the "Assigned hot-key" prompt, press the function or control key(s) to which this keystroke macro should be assigned. This is the key(s) that you will later press to play back the macro. Typically a keystroke macro is assigned to a key, such as **<F12>**, **<Shift-F10>** or **<Ctrl-T>**, which is currently unused or whose current assignment you want to overwrite.

   If necessary, you can edit the "Assigned hot-key" sequence with **[BACK-SPACE]**.
3. When "Assigned hot-key" is correct, press **<Tab>**. (DOS version: press **<Enter>**.)

   If the "Assigned hot-key" is already in use, you will be prompted for confirmation to overwrite the existing assignment.
4. For "Edit sequence:", press the exact sequence of keystrokes you want the macro to play back each time its hot-key is pressed.

   If necessary, you can edit the "Edit sequence" with **[BACKSPACE]**.
5. When the "Edit sequence" is correct, press **<Enter>**.

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, select **{CONFIG, Keyboard layout, Save layout}**.

Notes:

To use **<Esc>**, **<Enter>** or **<Backspace>** as part of the "Assigned hot-key" or "Edit sequence", you must precede that key with **[ENTER CTRL]** (default: **<Ctrl-Q>**). If it contains **[ENTER CTRL]**, press it twice.

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".

DOS, UNIX, QNX versions:

This item uses the edit functions **[ESCAPE]** and **[BACKSPACE]** which are almost always assigned to **<Esc>** and **<Backspace>**. 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.

See Also:

"Keystroke Macros" in Chapter 4 for a step-by-step example.
Notes for **{CONFIG, Keyboard layout}** above.
{CONFIG, Keyboard layout, Record keystroke macro}, **{CONFIG, Keyboard layout, Edit/view layout}**

Record Keystroke Macro

Define a keystroke macro by recording your editing operations.

Keystroke Equivalent:

<Alt-K>, This is a keystroke macro.

Full Description:

This item records a new keystroke macro while you are editing. This makes it easy to record a sequence of editing steps that you are going to use over and over again.

This is an alternative to **{CONFIG, Keyboard layout, Add keystroke macro}** and has the advantage of letting you see the editing operations as you make them. 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 Keystroke 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 that access the VEDIT PLUS macro language.

➢ **To record a new keystroke macro:**

1. Select **{CONFIG, Keyboard layout, Record macro}**.
2. At the "Assigned hot-key" prompt, press the function or control key(s) to which the recorded keystroke macro should be assigned. This is the key(s) that you will later press to play back the macro. Typically a keystroke macro is assigned to a key that is currently unused or whose current assignment you want to overwrite.

   If necessary, you can edit the "Assigned hot-key" sequence with **[BACKSPACE]**.
3. When "Assigned hot-key" is correct, press **<Tab>**. (DOS version: press **<Enter>**.)

   If the "Assigned hot-key" is already in use, you will be prompted for confirmation to overwrite the existing assignment.

4. VEDIT is now in "Record mode" — this is indicated on the status line. Everything you now type becomes part of the recorded keystroke macro.
5. To finish recording the new keystroke macro, press the "Stop Record" key which is displayed on the left hand side of the status line.

The "hot-key" for this menu item (default **<Alt-K>**) is the "Stop Record" key. When no "hot-key" is defined, the "Stop Record" key is **<Ctrl-]>** which is available on all keyboards.

See Also:

For the most part, all commentary and notes for **{CONFIG, Keyboard layout, Add keystroke macro}** apply here too.

Edit/view layout

Edit, view or print the keyboard layout.

Keystroke Equivalent:

(None)

Full Description:

This item lets you view or edit the entire keyboard layout as a normal text file. You can also print the layout and "cut and paste" between different layouts.

When done, the new layout is automatically loaded, and can optionally be made permanent by saving it as the **`vedit.key`** file.

Notes:

The topic "Editing the Keyboard Layout" in Chapter 4 describes editing the keyboard layout in more detail.

This function is implemented by the **`keyedit.vdm`** macro which completely controls its operation.

See Also:

"VEDIT.KEY Layout File" in Chapter 8.
{CONFIG, Keyboard layout, Load layout}

Display unused keys

Display a list of unused (unassigned) keys.

Keystroke Equivalent:

(None)

Full Description:

This item displays a list of keys that are currently unused — not assigned to an edit function or keystroke macro.

These keys are therefore available for use as "hot-keys" that can be assigned to menu items or other keystroke macros. These keys can be assigned with **{CONFIG, Keyboard layout, Add keystroke macro} {CONFIG, Keyboard layout, Record keystroke macro}**, or by editing the keyboard layout with **{CONFIG, Keyboard layout, Edit/view layout}**, or by directly editing the **`vedit.key`** file.

Notes:

The dialog boxes for "Add keystroke macro" and "Record keystroke macro" also let you view the list of unused keys.

See Also:

{CONFIG, Keyboard layout, Add keystroke macro}, **{CONFIG, Keyboard layout, Record keystroke macro}**
"Keystroke Macros" and "Editing the Keyboard Layout" in Chapter 4.
"VEDIT.KEY Layout File" in Chapter 8.

Save Layout to a File

Save entire keyboard layout in **`vedit.key`** or another file.

Keystroke Equivalent: (None)

Full Description:

This item saves the entire keyboard layout, including any new keystroke macros, into the **`vedit.key`** file or other specified file. By saving into **`vedit.key`**, any layout changes will be permanent (or until changed again).

When VEDIT starts up, it loads the keyboard layout from the **`vedit.key`** file.

➢ **To save the current keyboard layout:**

1. Select **{CONFIG, Keyboard layout, Save layout}**. You are prompted with:

   `Filename: C:\VEDIT\VEDIT.KEY`

2. If the default "vedit.key" filename is correct, press **<Enter>**. Otherwise edit the path and filename as needed, or enter "***.key**" for point & shoot file selection.

Since **`vedit.key`** normally already exists, you will be prompted for confirmation to overwrite it.

Notes:

The topic "VEDIT.KEY Layout File" in Chapter 8 (Configuration) describes how to edit the **`vedit.key`** file.

This functions does not save any comments and **Config( )** commands that you may have added to the **`vedit.key`** file.

See Also:

"VEDIT.KEY Layout File" in Chapter 8.
{CONFIG, Keyboard layout, Edit/view layout}, **{CONFIG, Keyboard layout, Load layout}**

Load Layout from a File

Load an entire keyboard layout from a .KEY file.

Keystroke Equivalent: (None)

Full Description:

This item loads an entire new keyboard layout from a **.key** file supplied by us or previously created with **{CONFIG, Keyboard layout, Save layout}.** This lets you change the entire keyboard layout "on the fly" and load different sets of keystroke macros for different editing tasks.

➢ **To load an entire new keyboard layout file:**

1. Select **{CONFIG, Keyboard layout, Load layout}**.
2. The default filename is "vedit.key". You can either select this name by pressing **<Enter>**, enter the desired filename, or enter "***.key**" for point & shoot file selection. The new keyboard layout table will be loaded.

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 with **{CONFIG, Keyboard layout, Save layout}** (But don't overwrite the file you are about to load.)

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. Typical errors are misspelling the name of a key or edit function.

See Also:

{CONFIG, Keyboard layout, Edit/view layout}, **{CONFIG, Keyboard layout, Save layout}**

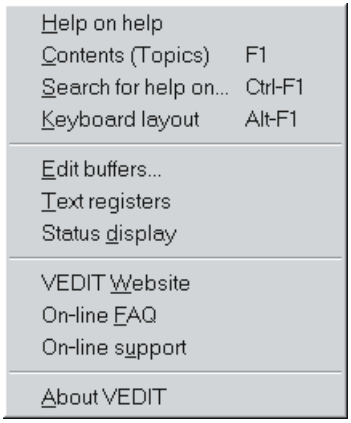
Help Menu

Keystroke Equivalent:

<Alt-H>, This is a keystroke macro.

Introduction:

The Help menu provides access to the on-line help "Table of Contents", the basic keyboard layout, information about the text registers and edit buffers that are in use, and information related to the current status of VEDIT.



Help on Help

General help on using Windows on-line help.

Keystroke Equivalent: (None)

Full Description:

This item starts up the general help topic on how to use a Windows program's on-line help. This is the same information displayed by other programs that have a "Help on help".

In other words, this feature is built into Windows; we have no control over it.

Contents (Topics)

Activate VEDIT's on-line help system; displaying the Table of Contents.

Keystroke Equivalent:

<F1> while no menu or dialog box is displayed; this is the **[HELP]** function.

Full Description:

VEDIT provides an extensive on-line help system. This item enters the on-line help and starts at the "Table of Contents".

You can then select any desired topic and navigate through the on-line help in the usual manner. You can use the "<<" and ">>" buttons to navigate to the previous and next topics.

DOS, UNIX, QNX Versions Only:

The non-Windows versions have an on-line help system that is unique to VEDIT.

➢ **To access help on using the on-line help:**

1. Select **{HELP, Contents}** (default: **<F1>**).
2. Press **[HELP]** (default: **<F1>**).

See Also:

"Context Sensitive Help" in Chapter 3.

Keyboard layout

Display the basic keyboard layout.

Keystroke Equivalent:

<Alt-F1>, This is a keystroke macro.

Full Description:

This item displays the basic keyboard layout. Each *edit function* is shown along with the keys assigned to it.

The basic edit functions are described in Chapter 7 (Edit Function Reference).

The keyboard layout can be changed with **{CONFIG, Keyboard layout, Edit/view layout}** or by editing the **`vedit.key`** file directly as described in Chapter 8 (Configuration).

DOS, UNIX, QNX Versions Only:

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

The display of the built-in keystroke macros is only applicable to the "Normal" keyboard layout. If you are using a different layout, you can edit the topic "NORMALK" in the on-line help file. Modifying the on-line help file is described in the on-line help topic "ONLINE".

This item is equivalent to entering the on-line help system and selecting **[K]ey-layout**.

Edit Buffers

Displays edit buffer information and permits "point and shoot" selection of any buffer.

Keystroke Equivalent: (None)

Full Description:

This item is identical to **{FILE, Buffer switch}**.

This item displays the ID numbers of all open edit buffers and lists which window(s) they are attached to. Buffers that have never been displayed in a window may not be attached to any window. (The VEDIT PLUS macro language can create unattached buffers and explicitly attach/detach them.)

It also displays the full pathname of the files being edited, if any. The filename is preceded with "*" if the file has been altered. It is preceded with "!" if the file is in Read-only mode.

If you are only viewing the buffer information, press **<Esc>** or select **[Cancel]** to return to your editing when done. Alternatively you can switch to another edit buffer.

Notes:

This item duplicates **{FILE, Buffer switch}** so that all status information can be found in one menu.

See Also:

{FILE, Buffer switch}, **{FILE, Open}**

Text Registers

Display active text registers: their name, size and contents.

Keystroke Equivalent: (None)

Full Description:

This item displays the names of non-empty text registers, their size and up to the first 60 bytes of their contents.

See Also:

{HELP, Edit buffers}, **{FILE, Buffer switch}**

Status Display

Display VEDIT's version, release date and status information.

Keystroke Equivalent: (None)

Full Description:

The VEDIT Status Display provides important information about the copy of VEDIT you are using and its present state. Here is an example:

The following information is displayed:

Product name, version and release date. You will need this information when you contact Greenview Data, Inc. 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 by selecting **[ ] Change directory** in any file selection dialog box.
Input file	The file VEDIT last opened for editing. It is the same as the "Output File" unless you selected **{FILE, Save as}** to save the file under a different name.
Output file	The file into which VEDIT will save the editing changes.
Output file size	This would be the size of the output file if you saved it right now.
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" (auto-buffer) some of the file to a temporary disk file to keep this value somewhat constant. The total memory space available to VEDIT is typically much larger. It is used to hold other files, text registers, etc.
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 the actual size of your file.
Used in T-Regs	This is the number of characters that are currently stored in all text registers. The maximum value is about 60,000 characters, but may be less if memory is limited. You may want to empty any large, unneeded text registers.
Date	The current date as reported by your computer. If this is not correct, refer to your computer system documentation for information on how to set the current date.
Time	The current time as reported by your computer.

VEDIT Website

On-line FAQ

On-line support

Quickly access the VEDIT Web site.

Keystroke Equivalent: (None)

Full Description:

These items start your Internet Web browser and connect to the extensive VEDIT Website. The Home page lists the most current version of VEDIT PLUS.

Note:

These functions only work under Windows 95/98/NT and require that Web addresses are associated with your Web browser, as is typical with Netscape (tm) and Internet Explorer (tm).

You can also receive support by sending e-mail to **support@vedit.com**.

About

Display the VEDIT version number, copyright and technical support telephone number.

Keystroke Equivalent: (None)

Full Description:

This item displays the VEDIT version number and release date. The Windows version also displays the Serial number. You will need this information when contacting Greenview Data, Inc. for technical support or upgrade assistance. It describes exactly which product and version you are using.

This item also displays the current technical support telephone number for VEDIT and our Internet addresses.

Note:

The Windows version also displays through what month and year you can receive technical support and download updates from our Website. For example:

Support expires at end of 10-2000

This indicates that you can download and install a new VEDIT PLUS dated, for example, 15-Oct-2000. However, you would need to purchase an update to a VEDIT PLUS dated 10-Nov-2000. Our Website always lists the current VEDIT PLUS' version number and release date.

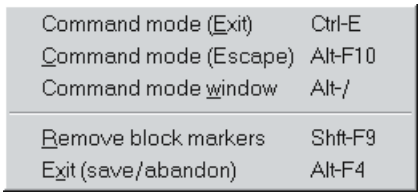
Escape Menu

Keystroke Equivalent:

<Esc>, This is the edit function **[ESCAPE]**

Introduction:

The Escape menu provides quick access to several commonly used functions. These items have been selected because they all represent a conventional and logical use of the **<Esc>** key; that is, they let you exit, or "escape" from a particular common situation.



To select the **{ESCAPE}** menu, press **<Esc>**. Since this key is also used to exit, or "escape from" the menu system, dialog boxes and other prompts, it will only bring up the **{ESCAPE}** when no menu or dialog box is displayed.

Press **<Esc>** again to escape from the **{ESCAPE}** menu.

Notes:

Technically, it is the **[ESCAPE]** function that displays this menu. However, since **<Esc>** is always assigned to **[ESCAPE]** under Windows and DOS, we use the two interchangeably. However, under UNIX and on CRT terminals, you may have to press the **<Esc>** key twice.

Command Mode (Exit)

Command Mode (Escape)

Exits or "escapes" Visual Mode to the Command Mode and possibly the "COMMAND:" prompt.

Keystroke Equivalent:

<Ctrl-E>, This is identical to the edit function **[VISUAL EXIT]**

<Alt-F10>, This is identical to the edit function **[VISUAL ESCAPE]**

Full Description:

"Command Mode (Exit)" exits Visual Mode and enters Command Mode. Any command macro which is currently running will continue to run. If no command macro is running, you will receive the "COMMAND: " prompt.

"Command Mode (Escape)" exits Visual Mode and aborts any command macro which is running. It will normally give you the "COMMAND:" prompt. However, if a "locked-in" command macro is running, it will restart that macro.

If the special "<**$**>" Command Mode window exists, VEDIT will switch to this window to display the "COMMAND:" prompt. Otherwise, the prompt will be displayed at the bottom of the current window.

See Also:

"Basics - Entering Command Mode" in Chapter 2 (Command Mode Guide) of the VEDIT PLUS Reference Manual.

Command Mode Window (VEDIT PLUS only)

Create the special Command Mode window and switch to it.

Keystroke Equivalent:

<**Alt-/**>, This is a keystroke macro.

Full Description:

This function creates the special Command Mode window "$" as a 5-line reserved window at the bottom of the screen and then performs a "Command mode (Escape)". Unless a "locked-in" macro is running, you will get the "COMMAND:" prompt.

Pressing this function's hot-key (default: <**Alt-/**>) at the COMMAND: prompt deletes the Command Mode window and returns to Visual Mode.

HINT: This hot-key (default: **<Alt-/>**) is a convenient way to toggle between the Visual Mode and the Command Mode.

See Also:

Chapter 2 in the VEDIT PLUS manual.
On-line help topic "Command Mode Basics" (DOS: "CMD").

Remove Block Markers

Remove any existing block markers.

Keystroke Equivalent:

<**Shft-F9**>, this is a keystroke macro.

Brief Description:

Remove any block markers that are currently set; this removes any block highlighting.

Notes:

It is usually easier to remove the block markers by pressing **[CANCEL]** (<**Ctrl-**\>) or <**Ctrl-Break**>. To remove the block markers with a mouse, simultaneously press both mouse buttons.

See Also:

Identical to **{BLOCK, Remove markers}**.

Exit (save/abandon)

Keystroke Equivalent:

<Alt-F4>, This is a keystroke macro.

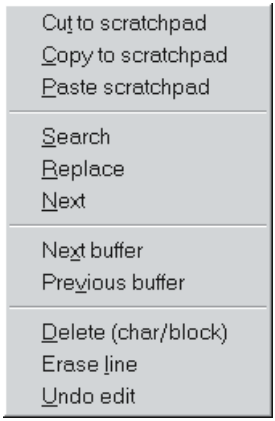
Brief Description:

Save the current file and exit VEDIT.

See Also:

Identical to **{FILE, Exit}**.

Mouse Right-Click Menu



Introduction:

When you press the right mouse button during normal editing, the Right-Click menu is displayed at the mouse position. It simply duplicates some of the more commonly used functions in the main menu.

Most of the Right-Click menu items also have equivalent "hot-keys" and are duplicated on the toolbar. Therefore, it is simply an alternative way of accessing these items. It is probably more useful when the toolbar is not displayed.

Cut to Scratchpad — Cut (move) the highlighted block to the scratchpad (text register 0). Same as **{EDIT, Scratchpad, Cut to scratchpad}**.

Copy to Scratchpad — Copy the highlighted block to the scratchpad. Same as **{EDIT, Scratchpad, Copy to scratchpad}**.

Paste Scratchpad — Paste (insert) the scratchpad at the cursor position. Same as **{EDIT, Scratchpad, Paste scratchpad}**.

Search — Start a new search. Same as **{SEARCH, Search}**.

Replace — Start a new search and replace. Same as **{SEARCH, Replace}**.

Next — Search or search & replace the next occurrence. Same as **{SEARCH, Next}**.

Next Buffer — Toggle round-robin fashion to the next buffer (file). Same as **{FILE, Next buffer}**.

Previous Buffer — Toggle round-robin fashion to the previous buffer (file). Same as **{FILE, Previous buffer}**.

Delete (Char/Block) — Delete the current character or highlighted block. Same as **{EDIT, Delete, Delete (char/block)}** or **[DELETE]**.

Erase Line — Delete the current line. Same as **{EDIT, Delete, Erase line}** or **[ERASE LINE]**.

Undo Edit — Undo the last editing keystroke. Same as **{EDIT, Undo, Edit}**.

Chapter 7

Edit Function Reference

Edit functions are the basic building blocks of VEDIT's operation. Each edit function is assigned to a specific key or key combination. Edit functions can also be assigned to several (duplicate) keys. For example, the "Normal" layout assigns **[T-REG INSERT]** to both **<Numpad*>** and **<F11>**.

Many basic editing functions, such as cursor movements, are also available in dialog boxes.

For more information on assigning keys to edit functions see **{CONFIG, Keyboard layout}** in Chapter 6 (Menu Reference) and Chapter 8 (Configuration).

You can display which key or keys are assigned to each edit function by selecting **{HELP, Keyboard layout}**.

[BACKSPACE]

Deletes the character to the left of the cursor. At the beginning of a line, it deletes the preceding "newline" character, effectively merging the lines together.

This function's behavior is configurable with **{CONFIG, Emulation, [BACKSPACE] emulation mode}**.

In dialog boxes, it permits editing the current entry.

In Window's dialog boxes, this function is always **<Backspace>**.

[BACKTAB]

Moves the cursor to the character at the previous tab stop. It stops at the beginning of a line.

If the cursor is in a highlighted block, it undents all lines in the block, same as **{EDIT, Undent}**. This behavior can be changed with **{CONFIG, Emulation, [TAB CHARACTER] emulation mode}**.

In dialog boxes, it always moves to the previous item.

[CANCEL]

During normal editing (no prompts) it removes any block markers that are set. Also stops any **[REPEAT]** operation.

DOS, UNIX, QNX versions: It also cancels the menu system and any dialog box. In Windows/DOS, pressing **<Ctrl-Break>** performs **[CANCEL]**.

It is assigned to **<Ctrl-\>** in most keyboard layouts.

See also: **{BLOCK, Remove markers}** in Chapter 6.

[CURSOR UP]

Moves the cursor up one line, to the same horizontal position. 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. The cursor cannot be moved past the last line in the file.

[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 configurable with **{CONFIG, Emulation, Cursor positioning mode}**. Set to "4", the cursor can move past the end of a line.

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; this behavior is configurable with **{CONFIG, Emulation, Special emulation modes}**.

[DELETE]

Deletes the character at the cursor. At the end of a line, it deletes the "newline". (The Windows/DOS "newline"consists of the Carriage-Return and Line-Feed characters.)

If a block is currently highlighted and the cursor is within the block (or immediately past it), it deletes the block, same as **{BLOCK, Edit/Translate, Block delete}**.

This function's behavior with respect to "newlines" and blocks is configurable with **{CONFIG, Emulation, Special emulation modes}**.

See also: **{EDIT, Delete, Delete}** in Chapter 6.

[DEL PREV WORD]

Deletes the word, or portion of a word, or whitespace to the left of the cursor. Pressing it again deletes the next whitespace or word.

[DEL NEXT WORD] Delete the word, or portion of a word, to the right of the cursor.

[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.

It is assigned to **<Ctrl-Shift-^>** in most keyboard layouts. Windows and DOS versions: it is also assigned to **<Ctrl-Q>**.

See also: "Keyboard Characters and Screen Display" in Chapter 4; **{EDIT, Enter CTRL char}** and **{MISC, ASCII table}** in Chapter 6.

[ERASE BOL] Erases (deletes) all characters from the beginning of the line up to the cursor.

See also:**{EDIT, Delete, Erase BOL}** in Chapter 6.

[ERASE EOL] Erases (deletes) all text from the cursor to the end of the current line.

See also:**{EDIT, Delete, Erase EOL}** in Chapter 6.

[ERASE LINE] Erases (deletes) the entire current line.

See also:**{EDIT, Delete, Erase line}** in Chapter 6.

[ESCAPE] "Escapes" from the current prompt or menu level. If there is no prompt or menu, it pops up the **{ESCAPE}** menu.

See also: "Using Menus" in Chapter 4; {ESCAPE} Menu in Chapter 6.

[HELP] Displays context sensitive help relevant to the current menu or prompt. If there is no prompt or menu, it starts up the on-line help at the Table-Of-Contents, same as **{HELP, Topics}**. When the menu system or a prompt is on the screen, **[HELP]** directly access the topic relevant to the currently displayed menu or prompt.

See also: **{HELP, Topics/Contents}** in Chapter 6.

[INSERT TOGGLE] Toggles between "Insert" and "Overstrike" modes.

See also: **{EDIT, Delete, Insert mode}** in Chapter 6.

[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. This function's behavior is configurable with **{CONFIG, Emulation, [LINE BEGIN/END] 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. This function's behavior is configurable with **{CONFIG, Emulation, [LINE BEGIN/END] 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.

In the Windows and DOS versions, this function is also activated by just tapping the **<Alt>** key.

[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 6.

[NEXT WORD] Moves 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. The amount of screen overlap is configurable with **Config(S_PG_OVERLAP)** - see Chapter 8.

[PAGE DOWN] Moves the cursor to the next screen "page" — similar to pressing **[CURSOR DOWN]** for 3/4 screen lines.

[PREV 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.

See also: "Word Processing Function - Definition of Paragraph" in Chapter 4.

[PREV 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] This function is identical to **{EDIT, Repeat}**.

[REPEAT LAST] Repeats the last typed character, edit function, menu selection or keystroke macro.

[REPEAT LAST] is often used to repeat menu selections for which there is no hot-key.

See also: **{EDIT, Repeat}** in Chapter 6.

[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 (**<CR><LF>** pair). Pressed in the middle of a line, it splits the line. Its behavior can be changed with **{CONFIG, Emulation, <Enter> key emulation mode}**.

[SCREEN BEGIN]

Moves the cursor to the beginning of a line at or near the top of the current window (screen). This function's behavior is configurable with **{CONFIG, Emulation, Special emulation modes}**.

[SCREEN END]

Moves the cursor to the end of a line at or near the end of the current window (screen). This function's behavior is configurable with **{CONFIG, Emulation, Special emulation modes}**.

[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. Its behavior can be changed with **{CONFIG, Emulation, Special emulation modes}**.

See also: "Scrolling the Screen" in Chapter 4.

[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, Display options, 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. Its behavior is configurable with **{CONFIG, Emulation, [TAB CHARACTER] emulation mode}**.

If the cursor is in a highlighted block, it indents all lines in the block, same as **{EDIT, Indent}**.

In dialog boxes, it always moves to the next item.

See also: "The <Tab> Key and Tab Characters" in Chapter 4; **{CONFIG, Tab/Fill, Expand <Tab> with spaces}** in Chapter 6.

[T-REG COPY]
[T-REG MOVE]
[T-REG INSERT]

These functions are identical to **{BLOCK, Copy to register}**, **{BLOCK, Move to register}** and **{BLOCK, Insert register}**.

At the "COMMAND:" prompt, **[T-REG INSERT]** inserts the contents of the scratchpad (text register 0).

[VISUAL ESCAPE]
(VEDIT PLUS only)

Exits the "Visual Mode" and aborts any command macro, such as WILDFILE.VDM, that is currently running.

In VEDIT, if no command macro is currently running, this function has no effect.

In VEDIT PLUS, this function is normally used to enter the "Command Mode" and is identical to **{ESCAPE, Command Mode (Escape)}**.

[VISUAL EXIT]

This edit function is used inside keystroke macros that access the VEDIT PLUS macro language.

It also permits exiting the "Visual Mode" so that a command macro, such as WILDFILE.VDM, can continue running.

In VEDIT PLUS, it enters the "Command Mode" when no macro is currently running. It is identical to **{ESCAPE, Command Mode (Exit)}**.

Chapter 8
Configuration

VEDIT is completely configurable — over 200 parameters and the entire keyboard layout can be configured to your precise needs and personal preferences.

NOTES: Please don't make major changes to VEDIT's configuration until you are familiar with it and understand what you are changing. Otherwise you will change VEDIT so much that it no longer works as described in the manual and on-line help.

The DOS and QNX version configuration is slightly different. Refer to the on-line help topic "CFG" for details.

All common configuration changes can be made with the **{CONFIG}** menu. The keyboard layout can be changed and new keystroke macros added with the **{CONFIG, Keyboard layout}** sub-menu.

These changes can either be temporary or permanent. Temporary means that the changes are lost when you exit VEDIT. Permanent means that the changes will be there the next time you run VEDIT. (We call it permanent, but of course you can change them again.)

To make configuration and keyboard changes permanent, they must be saved into the **`vedit.cfg`** and **`vedit.key`** files. VEDIT automatically loads these files on startup and thereby configures itself.

As you become more familiar with VEDIT, you may prefer to make configuration and keyboard layout changes by editing the **`vedit.cfg`** and **`vedit.key`** files directly. Some additional (rarely used) configurations can only be made by editing the **`vedit.cfg`** file.

SUGGESTION: You may want to save a copy of your **`vedit.cfg`** and **`vedit.key`** files in another directory, e.g. **`c:\vedit\save\vedit.cfg`**. Your preferred VEDIT configuration and keyboard layout can then be restored if you, or someone else, makes unwanted changes to it.

Basic Configuration

Most configuration changes are made by selecting items in the **{CONFIG}** menu. To save the changes, you must also select **{CONFIG, Save config}** and choose the default filename of "vedit.cfg" in the *User Config Directory*, typically **`c:\vedit\save\vedit.cfg`**.

Alternatively, if **{CONFIG, Auto-save config}** is enabled, VEDIT automatically saves configuration changes as you make them. VEDIT is supplied with "Auto-save config" enabled, but after you are familiar with VEDIT, you may decide that you don't want to automatically save every configuration change.

Similarly, the keyboard layout can be changed and new keystroke macros added with the **{CONFIG, Keyboard layout}** sub-menu. To save the changes, you must also select **{CONFIG, Keyboard layout, Save layout}** and choose the default filename, typically **`c:\vedit\save\vedit.key`**.

NOTES: VEDIT does not automatically save changes to the keyboard layout. You must select **{CONFIG, Keyboard layout, Save layout}**.

DOS Version: The basic configuration described in the manual and on-line help assume that **{CONFIG, Misc, Auto-load config}** is set to "3".

How VEDIT Configures Itself

To fully understand (and possibly troubleshoot) VEDIT's configuration, you need to know exactly how VEDIT configures itself at startup.

Note: The DOS and QNX version configuration is somewhat different. Refer to the on-line help topic "CFG" for details.

1. The executable VEDIT file (e.g. **`vpw.exe`**) contains the complete "default" configuration and the "normal" keyboard layout.

   DOS version: You can change this built-in configuration by selecting **{CONFIG, Misc, Save into VEDIT.EXE}**.

   If no other configuration files are found, or the "**-ixxx -g** " invocation options are specified, this determines the startup configuration.

2. VEDIT opens the **`vedit.ini`** file which it expects to find in the same directory as the executable **`vpw.exe`** file. This is usually the directory into which you installed VEDIT.

   The entries "HomeDir" and "UserCfgDir" determine the *VEDIT Home Directory* and *User Config Directory* which are used to locate other files.

   For most installations, the location of the **`vpw.exe`** and **`vedit.ini`** files, the *VEDIT Home Directory* and the *User Config Directory* will all be the same, e.g. **`c:\vedit`**. However, for shared network installations, they might be different. (See the Chapter 2 or the on-line help topic "Network Installation" for details.)

Details: VEDIT first opens the **vedit.ini** file in the same directory as the executable **vpw.exe**. It then reads the "UserCfgDir" value and opens the **vedit.ini** file in the *User Config Directory* as the working "vedit.ini" file. For most installation, there is only one **vedit.ini** file. However, for shared network installations, they are two **vedit.ini** files; the **vedit.ini** file on the network server uses "UserCfgDir" to specify the location of the user's **vedit.ini** file.

3. VEDIT searches the current directory and then the *User Config Directory* (typically **c:\vedit**) for the configuration files **vedit.cfg** and **vedit.key**. If found, they override the configuration and keyboard layout built into **vpw.exe**. No error is given if these files are not found.

   If the "**-g**" invocation option was specified, this step is skipped.

   Note that **vedit.key** must contain the complete keyboard layout. However, you could manually edit **vedit.cfg** to have only a few configuration parameters; the other parameters would come from **vpw.exe**.

4. VEDIT searches first the current directory, then the *User Config Directory* and finally the *VEDIT Home Directory* for the **startup.vdm** file. This is a macro (written in the VEDIT PLUS macro language) that further configures VEDIT on startup.

   startup.vdm is primarily used to set up the **{TOOLS}** and **{USER}** menu, file-type specific configuration, color syntax highlighting and template editing.

   The supplied **startup.vdm** file does not override any configuration settings or change the keyboard layout. However, it documents how you can override any configuration settings and add some useful keystroke macros to the keyboard layout.

   Note: The name "startup.vdm" is actually specified by the "Startup" entry in the **vedit.ini** file. Advanced users could change it, but all VEDIT documentation uses the name "startup.vdm".

5. VEDIT can optionally configure itself according to the filename extension of each file being edited. For example. ".txt" files can have word processing features enabled, while ".c" files have programming features and color syntax highlighting enabled.

   The supplied **startup.vdm** sets up this file-type specific configuration feature, but does not enable it. You must enable it with **{CONFIG, File handling, File-type specific configuration}** and then select **{CONFIG, Save config}**. Alternatively, it can be enabled by editing **startup.vdm**.

   This feature typically changes the tabs stops and the settings in the **{CONFIG, Word processing}** and **{CONFIG, Programming}** submenus according to the file type.

Troubleshooting

If your configuration changes are lost the next time you run VEDIT, these troubleshooting steps can help you solve the problem:

1. Make sure that you really did save your configuration changes by selecting **{CONFIG, Save config}**. Most configuration changes are automatically saved if **{CONFIG, Auto-save config}** is enabled.

   Changes to the keyboard layout must be saved with **{CONFIG, Keyboard layout, Save layout}**; they are not automatically saved.

   To verify that the configuration changes were saved, open and examine the files **`vedit.cfg`** and **`vedit.key`** in the *User Config Directory*, e.g. **`c:\vedit\vedit.cfg`**. You can directly edit these files.

2. Check that your **`vedit.ini`** file is correct. Open the **`vedit.ini`** file in the *User Config Directory*. For non-network installations, this is the directory into which you installed VEDIT; it is typically **`c:\vedit`**.

   The **`vedit.ini`** file should be fully documented. The first line should read something like:

   ; VEDIT.INI - Windows information file for VEDIT.

   If the file is not documented, e.g. the first line reads "[VEDIT]", this indicates that VEDIT did not find a complete **`vedit.ini`** file and therefore auto-created one. However, the auto-created **`vedit.ini`** is not complete and likely to cause problems. In this case you should copy the **`vedit.ini`** file from the original VEDIT disk (or downloaded .ZIP file) to your VEDIT directory.

   As described below, be sure that "HomeDir" and "UserCfgDir" specify the correct *VEDIT Home Directory* and *User Config Directory*.

   Also be sure that the entry "Startup" is set to the filename "startup.vdm".

   VEDIT PLUS: Go to the Command Mode "COMMAND:" prompt and enter the command "**Config_String( )**" to verify the current settings for *VEDIT Home Directory* and *User Config Directory*.

3. Make sure that the "UserCfgDir" entry in the **`vedit.ini`** file specifies the correct *User Config Directory*; it is typically **`c:\vedit`**.

   Then check that this directory contains the files **`vedit.cfg`** and **`vedit.key`**. Remember, **{CONFIG, Save config}** creates the **`vedit.cfg`** file, and **{CONFIG, Keyboard layout, Save layout}** creates the **`vedit.key`** file.

4. Make sure that the "HomeDir" entry in the **`vedit.ini`** file specifies the correct *VEDIT Home Directory*; it is typically also **`c:\vedit`**.

   For most installations, the *VEDIT Home Directory* and *User Config Directory* are set the same, e.g. **`c:\vedit`**. However, if VEDIT is installed on a shared network server for multiple licensed users, the *VEDIT Home Directory* is typically set to the network server, e.g. "h:\apps\vedit"

and the *User Config Directory* is typically set to a local hard disk, e.g. “c:\vedit”.

If the *User Config Directory* is inadvertently set to the network server, then your startup configuration will be the configuration saved by the last VEDIT user.

5. The **`startup.vdm`** file may be overriding your saved configuration changes. Try starting VEDIT without the **`startup.vdm`** by using the invocation option:

   vpw -ixxx

   The “-ixxx” attempts to load the file “xxx”, which presumably does not exist, in place of the **`startup.vdm`** file.

   If your configuration changes are still lost, open the file **`vedit.cfg`** in the *User Config Directory* and verify your configuration. If desired, you can directly edit this file.

   If your configuration changes are now restored, try setting **{CONFIG, File handling, File-type specific configuration}** to “0”. Select **{CONFIG, Save config}** to be sure the change is saved. Then start up VEDIT normally. If your configuration is now correct, you may want to study the file-type specific configuration section of the **`startup.vdm`** file. If your configuration is still lost, some other commands in **`startup.vdm`** are overriding it.

See also:

The topic “Startup.vdm File” in Chapter 5 (Advanced Topics).

The topic “Network Installation” in Chapter 2 (Installation).

VEDIT.KEY Layout File

The keyboard layout can be changed by editing the **`vedit.key`** file directly or by selecting **{CONFIG, Keyboard layout, Edit/view layout}**. The latter gives you the choice of making the layout changes temporary or saving them into **`vedit.key`**.

There are several reasons for always editing the **`vedit.key`** file directly and ***never*** saving it with **{CONFIG, Keyboard layout, Edit/view layout}** or **{CONFIG, Keyboard layout, Save layout}**:

- You can add comments to the layout for future reference. (Comments are lost if **{CONFIG, Keyboard layout}** saves the layout.)
- You can add **Config( )** commands to the **`vedit.key`** file to force any configuration changes needed to make the keyboard layout work properly. (**Config( )** commands are lost if **{CONFIG, Keyboard layout}** saves the layout.)

After creating a custom keyboard layout, you may want to save it not only as **`vedit.key`**, but use **{FILE, Save as}** to save it under an additional name, such as "bobs.key".

A new keyboard layout can be loaded at any time by selecting **{CONFIG, Keyboard layout, Load layout}**. Any supplied ".key" file such as **`normal.key`**, **`brief.key`** or **`wordstar.key`** can be loaded. A custom file, such as "bobs.key" can also be loaded.

NOTE: After carefully personalizing and commenting a **`vedit.key`** file, be sure to save a copy under another name or in another directory. In particular, don't overwrite it with **{CONFIG, Keyboard layout, Save layout}**, because any comments and **Config( )** commands will then be lost.

➢ **To edit the `vedit.key` file:**

1. Select **{FILE, Open}** (default: **<Ctrl-O>**) and select the file in the *User Config Directory*, typically **`c:\vedit\vedit\vedit.key`**.
2. Change the layout as described below.
3. Save your changes, e.g. with **{FILE, Save and continue}** or **{WINDOW, Close}**.
4. Select **{CONFIG, Keyboard layout, Load layout}** to load the new **`vedit.key`** file and verify that the layout is correct.

   The next time VEDIT is started, it should have the new layout. If it does not, see "Troubleshooting" above; most likely, the startup macro **`startup.vdm`** is overriding the layout.

Modifying the VEDIT.KEY file

Follow these guidelines when editing the keyboard layout, either by editing the **`vedit.key`** file directly or by selecting **{CONFIG, Keyboard layout, Edit/view layout}**.

1. Normally, leave the first line alone. It assigns the **<Enter>** key to the **[RETURN]** function. It must be the first line of the keyboard layout. However, it can be preceded by comments or **Config( )** commands.
2. Notice that all remaining lines have the same format. Each line begins with the key or keys that are pressed to perform an editing function. This is followed by whitespace; at least a tab or two spaces. Then comes the entire editing sequence on one line.
3. 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" (i.e. Carriage-Return and Line-Feed) enter "[RETURN]".
4. You can add comment lines to the file by starting the line with two slashes "//". See the NOTE: above.

Configuration Commands in ".KEY" Files

A ".key" keyboard layout file (in Text mode), can optionally have any desired number of lines consisting of **Config( )** and other macro language commands. This is useful for setting up configuration parameters that are needed to accurately emulate other editors. For example, our supplied **`brief.key`** has the command **Config(E_Line_Mode,5)** to properly make the **<Home>** and **<End>** keys emulate the Brief (tm) editor.

When adding macro language commands to a **`vedit.key`** file, follow these guidelines:

- Each command line must begin in column 1 with a valid command. Comment lines can also be included beginning with the normal "//".
- The maximum line length for commands and comments is 1000 characters.
- Complex macros using flow control statements are supported; however each line must be a complete macro. Text register 119 is reserved specifically for use by macros in ".key" files.
- **{CONFIG, Keyboard layout, Save layout}** does not save any **Config( )** commands or comments back into the ".key" file.

Therefore, you may find it better to make all permanent keyboard layout changes by editing your personal ".key" file, e.g. "bobs.key", and then copying this file to **`vedit.key`**.

NOTE: Loading a new keyboard layout may not restore configuration settings set by a previous ".key" file.

VEDIT.CFG Configuration File

While most configuration changes can be made with the **{CONFIG}** menu, there are several reasons for editing the **`vedit.cfg`** file directly:

- All configuration parameters can be changed, including some not available in the **{CONFIG}** menu:
  - The additional characters that separate words from each other. The characters occurring at the beginning of a line that separate paragraphs from each other.
  - Number of lines of screen overlap when using **[PAGE UP]** and **[PAGE DOWN]**. Also the range of screen lines in which the cursor can move before the screen scrolls.
  - VEDIT PLUS: How many times the Command Mode help message is displayed.
  - Number of Undo levels available.
  - Enable or disable the "DOS shell" functions and the DOS prompt to be used when shelled out.
  - (DOS only) Force VEDIT to start up in 25, 28 or 50 line mode.
- Making numerous changes is often easier by editing the **`vedit.cfg`** file. You can search for items and cut and paste between files.
- When upgrading to future versions of VEDIT, you will use the **`vedit.cfg`** and **`vedit.key`** files to transfer your configuration to the new version.
- VEDIT PLUS users can change any configuration parameter on the fly in the "Command Mode". Since **`vedit.cfg`** really is a VEDIT PLUS command macro, it helps to become familiar with the configuration commands in it.

➢ **To edit the `vedit.cfg` file:**

1. Select **{FILE, Open}** (default: **<Ctrl-O>**) and select the file in the *User Config Directory*, typically **`c:\vedit\vedit\vedit.cfg`**.
2. Change the desired configuration parameters. This is described below. Some of the parameters are described in this chapter; the remainder are described under the **{CONFIG}** menu in Chapter 6 (Menu Reference).
3. Save your changes, e.g. with **{FILE, Save and continue}** or **{WINDOW, Close}**.
4. Select **{CONFIG, Load config}** to load the **`vedit.cfg`** file and verify that the configuration is correct.

   The next time VEDIT is started, it should have the new configuration. If it does not, see "Troubleshooting" above; most likely, the startup macro **`startup.vdm`** is overriding the configuration.

List of Config( ) Parameters

The **vedit.cfg** file consists of over 200 **Config( )** commands. Most of the **Config( )** commands change parameters which are also in the **{CONFIG}** menu and are described in Chapter 6 (Menu Reference) and in the on-line help for each **{CONFIG}** sub-menu. The format of **Config( )** command lines is:

Config(*name***,"***comment***",***value***)**

name	The name of the configuration parameter to change.
comment	A descriptive comment in double-quotes that is ignored by VEDIT. It is only for your reference; for most parameters it is identical to the name seen in the **{CONFIG}** menu.
value	The parameter's value as a decimal number. This is the only field that you should change.

This topic primarily describes the additional **Config( )** commands which have no **{CONFIG}** menu equivalent.

Config(B_HORZ, "Horizontal border character", 205)
Config(B_...)

DOS, UNIX and QNX versions only. These parameters determine the characters used to draw boxes around its menus and dialog boxes. They are not used by the Windows version.

The following parameters determine the colors used by VEDIT. Most only apply to the DOS, UNIX and QNX versions. Only the editing colors and the syntax highlighting colors apply to the Windows version.

The parameter names beginning with "**C_**" are used on color displays, the parameter names beginning with "**C_X_**" are used on monochrome displays.

Config(C_TEXT, "Color for edited text ", 30)

The color used for the edited text. This color may be changed for individual windows with **{MISC, Color Toggle}** (**<Alt-J>** keystroke macro).

Config(C_STAT, "Color for status line", 112)

Color of the filename, line and column numbers, and empty parts of the status line. Usually set to reverse video.

Config(C_STAT_M, "Color for status line messages", 112)

Color used for status line messages including "BLOCK", and "INS". They are usually displayed in reverse video. To make these messages stand out, make this color different from "Color for status line". This color is also used for the "continuation characters" used when wrapping long lines on the screen.

Config(C_WIN_B_M, "Color for active window border", 63)

Color of the border around the active window. It is preferable to pick a color different from the edited text.

Config(C_WIN_B, "Color for inactive window border", 48)

Color of the borders around all inactive windows. It should be different from "Color for active window border".

Config(C_EMPTY, "Color for empty screen", 113)

Color of the background (desktop) screen "behind" all windows. It is only visible when windows don't cover the entire screen.

Config(C_BAR, "Color for menu bar", 112)

Color of the main menu bar at the top of the screen. Usually set to reverse video.

Config(C_BAR_S, "Color for menu bar selection", 127)

Color of the letter that selects each main menu item. This color should be different from "Color for menu bar". (However, the same color as "Color for menu bar highlighting" can be used).

Also used to highlight option letters and function keys in many prompts and help messages.

NOTE: If the selection letter is not visible on a monochrome display, try adjusting the contrast and brightness controls. If this doesn't work, pick another attribute value; try "7" or "15".

Config(C_BAR_H, "Color for menu bar highlighting", 78)

Color of the menu item that is currently selected.

Config(C_MENU, "Color for menu", 48)

Color of the pull-down menus. Coordinate this color with "Color for menu selection" and "Color for menu highlighting".

Config(C_MENU_S, "Color for menu selection", 62)

Color of the letter that selects each pull-down menu item. This color should be different from "Color for menu". (However, the same color as "Color for menu highlighting" can be used).

Config(C_MENU_H, "Color for menu highlighting", 78)

Color of the pull-down menu item that is currently selected.

Config(C_MENU_I, "Color for inactive menu items", 56)

Override for items in the **{BLOCK}** and other menus that are currently inactive. If "0", it is the same as "Color for menu".

Config(C_PROMPT, "Color for simple prompts", 47)

Color of simple one-line prompts, such as the prompt for **{GOTO, Line #}**.

Config(CO_DIALOG, "Color for dialog boxes (override)", 0)

Override color for dialog boxes. If "0", it is the same as "Color for menu bar".

Config(CO_DIALOG_S, “Color for dialog selection (override)”, 0)

Override color for the letter that selects each dialog box item. If “0”, it is the same as “Color for menu bar selection”.

Config(CO_DLG_BUT, “Color for dialog buttons (override)”, 0)

Override color for the inactive buttons in a full dialog box. If “0”, it is the same as “Color for menu”.

Config(CO_ACT_BUT, “Color for active button (override)”, 0)

Override color for the active button in a full dialog box. If “0”, it is the same as “Color for menu highlighting”.

Config(CO_BLOCK, “Color for block highlighting (override)”, 46)

Override color for blocks being highlighted. If “0”, the reverse video of the text color is used. For example, ”71” will always display blocks as white text on a red background.

Config(CO_CURSOR, “Color for cursor (override)”, 113)

(IBM DOS only) Override color for the software cursor types 1 and 2 (**{CONFIG, Characters/Cursors, Cursor type}**). If “0”, the reverse video of the text color is used. For example, ”64” always displays a red cursor.

If “Color for text” is set to a bright color such as yellow (attribute 14), you may not like the automatic cursor color. You should then set this parameter to the desired cursor color.

Config(CO_ERASE, “Color for window erase (override)”, 0)

Override for that portion of an editing window where there is no text. If “0”, the same color as the text is used.

Setting an overriding color gives an unusual effect, but lets you clearly see trailing spaces at the ends of lines.

Config(CO_HELP, “Color for Help window (override)”, 0)

Override for pop-up help windows. If “0”, the same color as the current text is used.

Config(CO_LINE, “Color for cursor line (override)”, 78)

Override for the current line’s color when **{CONFIG, Display options, Highlight cursor line}** is enabled. If “0”, the cursor line is displayed in “bright” characters.

Config(C_RESERVE_W1, “Color for reserved words 1", 28)

Color of the syntax highlighting items that match the definition for “Reserved1”.

Config(C_RESERVE_W2, “Color for reserved words 2", 27)

Color of the syntax highlighting items that match the definition for “Reserved2”.

Config(C_RESERVE_W3, "Color for reserved words 3", 29)

Color of the syntax highlighting items that match the definition for "Reserved3".

Config(C_RESERVE_W4, "Color for reserved words 4", 31)

Color of the syntax highlighting items that match the definition for "Reserved4".

Config(C_SYMBOL, "Color for symbols", 23)

Color of the syntax highlighting items that match the definition for "Symbols".

Config(C_COMMENT, "Color for comments", 18)

Color of the syntax highlighting items that match the definitions for "Comment" or "Linecmt".

Config(C_STRING, "Color for strings", 26)

Color of the syntax highlighting items that match the definition for "Strings".

Config(C_NUMBER, "Color for numeric", 23)

Color of the syntax highlighting items that match the definition for "Numeric".

Most of the following **Config( )** commands apply to all versions of VEDIT. Most correspond to items in the **{CONFIG}** menus.

Config(E_CR_MODE, "Cursor positioning mode (0 - 4)", 1)

Same as **{CONFIG, Emulation, Cursor positioning mode}**.

Config(E_EXP_TAB, "Expand <Tab> with spaces (*)

Same as **{CONFIG, Tab/Fill, Expand <Tab> with spaces}**.

Config(E_TAB_MODE, "[TAB CHARACTER] emulation mode", 3)

Same as **{CONFIG, Emulation, [TAB CHARACTER] emulation mode}**.

Config(E_ENTER_MODE, "<Enter> key emulation mode (0 - 3)", 1)

Same as **{CONFIG, Emulation, <Enter> key emulation mode}**.

Config(E_BS_MODE, "[BACKSPACE] emulation mode (0 - 3)", 0)

Same as **{CONFIG, Emulation, [BACKSPACE] emulation mode}**.

Config(E_LINE_MODE, "[LINE BEGIN/END] emulation mode", 3)

Same as **{CONFIG, Emulation, [LINE BEGIN/END] emulation mode}**.

Config(E_SPEC_MODE, "Special emulation modes (0 - 255)", 0)

Same as **{CONFIG, Emulation, Special emulation modes}**.

Config(E_SHIFT_MODE, "Alt/Ctrl/Shift key shortcut modes", 7)

Same as **{CONFIG, Emulation, Alt/Ctrl/Shift key shortcut modes}**.

Config(E_E_SHFT_BK, "Enable <Shift> block marking", 1)

Same as **{CONFIG, Emulation, Enable <Shift> block marking}**.

Config(E_BM_MODE, "Block marker emulation mode (0 - 2)", 0)

Same as **{CONFIG, Emulation, Block marker emulation mode}**.

Config(E_ADV_CR_BK, "Advance cursor past block insert", 1)

Same as **{CONFIG, Emulation, Advance cursor past block insert}**.

Config(E_TRIM_BK, "Trim spaces after columnar operation (*)", 1)

Same as **{CONFIG, Tab/Fill, Trim spaces after columnar block operation}**.

Config(E_BK_FILL_C, "Block fill character (*) (0 - 255)", 32)

Same as **{CONFIG, Tab/Fill, Block fill character}**.

Config(E_SCRL_MARG, "Horiz. scroll margin (*) (0=Off)", 0)

Same as **{CONFIG, Display options, Horiz. scroll margin}**.

Config(E_SCRL_INC, "Horiz. scroll increment (*) (1 - 100)", 20)

Same as **{CONFIG, Display options, Horiz. scroll increment}**.

Config(E_INS_MODE, "Insert mode (Visual mode)", 1)

Determines whether VEDIT starts in "Insert" or "Overstrike" mode. Set by **[INSERT TOGGLE]**. Since this **Config( )** parameter is not saved into `vedit.cfg`, the `startup.vdm` file must be used to set this parameter.

Config(E_EMULATE, "Emulation (0=VEDIT, 2,3=vi, 4=QNX)", 0)

Reserved. Note: the vi emulation package originally supplied with VEDIT (PLUS) version 3.7 is no longer available due to the new macro language introduced in version 4.0.

Config(F_AUTO_SAVE, "Auto-save interval (0=Off, minutes)", 0)

Same as **{CONFIG, File handling, Auto-save interval}**.

Config(F_BACKUP_MODE, "Backup files (0=Off, 1=.BAK, 2=Dir)", 1)

Same as **{CONFIG, File handling, Backup files}**.

Config(F_OVER_MODE, "Overwrite-only mode (*) (0=Off)", 1)

Same as **{CONFIG, File handling, Overwrite-only mode}**.

Config(F_SAVE_SESS, "Save session in current directory", 0)

Same as **{CONFIG, File handling, Save session in current directory}**.

Config(F_VSWAP, "Use V-SWAP when entering DOS", 1)

(DOS only) Same as **{CONFIG, File handling, Use V-SWAP when entering DOS}**.

Config(F_F_BROWSE, "Enable fast browse mode (*)", 1)

Same as **{CONFIG, File handling, Enable fast browse mode}**.

Config(F_AUTO_F_TYPE, "Enable auto-file type (*)", 1)

Same as **{CONFIG, File handling, Enable auto-file type}**.

Config(F_F_TYPE, "File type (*) (0=CR+LF, 1=LF, 2=CR,)", 0)

Same as **{CONFIG, File handling, File type}**.

Config(F_REC_HEAD, "Record header size (*) (0 - 16K)", 0)

Same as **{CONFIG, File handling, Record header size}**.

Config(F_EOF_PROC, "Special E-O-F processing (*)", 0)

Allows VEDIT to create files compatible with old DOS programs (or other applications) that require a Ctrl-Z End-Of-File character.

If enabled, VEDIT does not read past the first Ctrl-Z when opening a file and will not insert Ctrl-Z characters. Three options control how files are written.

0 Normal text/binary file. Ctrl-Z is not treated specially.

1 Files are written in their exact file length. (Files are truncated at the first <Ctrl-Z> encountered.)

2 Files are written with one Ctrl-Z end-of-file marker.

3 Files are written with one Ctrl-Z and are padded with "nulls" to make the file length a multiple of 128.

Config(F_E_STARTUP, "Enable STARTUP.VDM file", 1)

(DOS only) A value of "1" enables the **`startup.vdm`** file. When set to "0" the **`startup.vdm`** is not loaded and executed. A change to this value must be saved into the VEDIT.EXE file with **{CONFIG, Misc, Save into VEDIT.EXE}**.

HINT: You can disable the **`startup.vdm`** file with the invocation option "**-i xxx**", where '**xxx**' is a non-existent file. No error message will be given. This is useful for debugging startup configuration problems.

Config(F_AUTO_BUF, "Enable auto-buffering (0, 1, 2)", 2)

This parameter should always be set to "2". Only provided for compatibility with older versions of VEDIT PLUS. It will be discontinued in a future version. If not set to "2", VEDIT will not automatically perform the file buffering necessary to edit files larger than memory!

Config(H_KEY_IN, "Keyboard input (0=ROM,1=System)", 0)

(DOS only) VEDIT can read the IBM PC keyboard via either the "ROM BIOS" or "System" calls. With "ROM BIOS", most keyboard enhancers are bypassed and have no effect on VEDIT's operation; via "System", they affect operation, which may be desirable or undesirable. (There is

little reason to use keyboard enhancers since VEDIT's keystroke macros are more flexible and better integrated.)

On some computers, when set to "System", some special keys such as the Numpad "*", "-" and "+" keypad keys, **<F11>** and **<F12>** are not available as function keys.

Config(H_KEY_RATE, "Keyboard typematic rate (0=Normal)", 0)

(DOS only) Same as **{CONFIG, Misc, Keyboard typematic rate}**.

The "**-k**" invocation option sets this to "0".

Config(H_POLL_MODE, "Keyboard polling (0=Compatibility)", 1)

(DOS only) Same as **{CONFIG, Misc, Keyboard polling}**.

The "**-j**" invocation option sets this to "0".

Config(H_USE_BIOS, "Write screen via BIOS", 0)

(IBM DOS only) VEDIT normally writes directly to the screen for maximum speed and flexibility. However, you can optionally configure VEDIT to write indirectly to the screen via the "BIOS". This is only needed for special applications. Screen display is significantly slower. A value of "2" permits the IBM PC version to run on some older non-IBMPC compatibles.

The "**-o**" invocation option overrides this setting.

Config(H_VGA_TYPE, "Enable VGA 28/50/30/34/60 lines (0 - 7)", 7)

(IBM DOS only) Same as **{CONFIG, Misc, Enable VGA 28/30/34/50/60}**.

Config(H_MOUSE_O, "Mouse options (0=Off, 1=Corner)", 1)

Same as **{CONFIG, Misc, Mouse options}**.

Config(P_PAPER_L, "Paper length (5 - 100)", 62)

Same as **{CONFIG, Printer, Paper length}**.

Config(P_TOP_MARG, "Top margin (0 - 60)", 3)

Same as **{CONFIG, Printer, Top margin}**.

Config(P_BOT_MARG, "Bottom margin (0 - 60)", 3)

Same as **{CONFIG, Printer, Bottom margin}**.

Config(P_LEFT_MARG, "Left margin (0 - 100)", 10)

Same as **{CONFIG, Printer, Left margin}**.

Config(P_RIGHT_MARG, "Right margin...", 0)

Same as **{CONFIG, Printer, Right margin}**.

Config(P_LINE_S, "Line spacing (1 - 4)", 1)

Same as **{CONFIG, Printer, Line spacing}**.

Config(P_DSP_MODE, "Print mode...", 0)

Same as **{CONFIG, Printer, Print mode}**.

Config(P_E_STRING, "Enable print job strings", 0)

Same as **{CONFIG, Printer, Enable print job strings}**.

Config(P_E_FF, "Enable Form-Feed", 1)

Same as **{CONFIG, Printer, Enable Form-Feed}**.

Config(P_E_PE, "Page eject on Finish/Eject", 1)

Same as **{CONFIG, Printer, Page eject on Finish/Eject}**.

Config(P_DEFAULT, "Printer (0=Default ... 7=File)", 0)

Same as **{CONFIG, Printer, Printer}**.

Config(PG_AUTO_IND, "Auto-indent mode (*)", 0)

Same as **{CONFIG, Programming, Auto-indent mode}**.

Config(PG_IND_INC, "Indent increment (*) (1 - 20)", 4)

Same as **{CONFIG, Programming, Indent increment}**.

Config(PG_CASE_CONV, "Lower/Upper case key conv. (*) (0 - 5)", 0)

Same as **{CONFIG, Programming, Lower/Upper case key conversion}**.

Config(PG_CONV_C, "Key conversion character (*) (32 - 126)", 59)

Same as **{CONFIG, Programming, Key conversion character}**.

Config(PG_F_AUTO_CONFIG, "File-type specific configuration", 0)

Same as **{CONFIG, Programming, File-type specific config}** .

Config(PG_TEMPLAT, "Enable template editing (*)", 0)

Same as **{CONFIG, Programming, Enable template editing}**.

Config(PG_E_SYNTAX, "Enable syntax highlighting (*)", 0)

Same as **{CONFIG, Programming, Enable syntax highlighting}**.

Config(S_STAT_OPT, "Status line display (1=Top, 2=Bottom)", 2)

(DOS only) Same as **{CONFIG, Display options, Status line display}**.

Config(S_H_CR_LINE, "Highlight cursor line (0 - 2)", 0)

Same as **{CONFIG, Display options, Highlight cursor line}**.

Config(S_DSP_MODE, "Screen display mode (0 - 32)", 0)

Same as **{CONFIG, Characters/Cursors, Screen display mode}**.

Config(S_NEWL_D_C, "Newline display character (*) (0 - 255)", 32)

Same as **{CONFIG, Characters/Cursors, Newline display character}**.

Config(S_TAB_D_C, "Tab display character (*) (0 - 255)", 32)

Same as **{CONFIG, Characters/Cursors, Tab display character}**.

Config(S_NULL_D_C, "Null display character (*) (0 - 255)", 0)

Same as **{CONFIG, Characters/Cursors, Null display character}**.

Config(S_E_MORE, "Enable -MORE- operation", 1)

Same as **{CONFIG, Display options, Enable -MORE- operation}**.

Config(S_CR_TYPE_O, "Cursor type in overstrike mode (0 - 6)", 2)

Same as **{CONFIG, Characters/Cursors, Cursor type in overstrike mode}**.

Config(S_CR_TYPE_I, "Cursor type in insert mode (0 - 6)", 1)

Same as **{CONFIG, Characters/Cursors, Cursor type in insert mode}**.

Config(S_CR_TYPE_V, "Cursor type in virtual space (0 - 6)", 0)

Same as **{CONFIG, Characters/Cursors, Cursor type in virtual space}**.

Config(S_CR_BLINK, "Cursor blink rate (2 - 25)", 8)

Same as **{CONFIG, Characters/Cursors, Cursor blink rate}**.

Config(S_CURSOR_C, "Cursor type-0 character (1 - 254)", 176)

Same as **{CONFIG, Characters/Cursors, Cursor type-0 character}**.

Config(S_ERASE_C, "Screen erase character", 32)

VEDIT normally clears the screen with spaces (value 32). However, for special applications, a different character could be used.

Config(S_EMPTY_C, "Empty (desktop) character", 176)

(DOS only) This character is displayed in the background (desktop) screen "behind" all windows. It is only visible when windows don't cover the entire screen. The color is set with **Config(C_EMPTY)**.

Config(S_N_LINE, "Number of screen lines (0=Auto)", 0)

(IBM DOS) With default value of "0", VEDIT starts up in the current number of screen lines. When set to other values, such as 28 or 50, VEDIT will attempt to switch the screen to the specified number of lines. This requires a VGA display adapter. Changes to this value must be saved into VEDIT.EXE with **CONFIG, Misc, Save into VEDIT.EXE}**.

(DOS CRT version only) Set this value to the number of lines on your CRT terminal; typically 24 or 25.

Config(S_N_COL, "Number of screen columns (0=Auto)", 0)

(DOS CRT version only) Set this value to the number of columns on your CRT terminal; typically 80 or 132.

Config(S_PG_OVERLAP, "Line overlap when paging", 2)

Determines the number of lines from the previous page that will be visible following a **[PAGE UP]** or **[PAGE DOWN]**. About 10% overlap is suggested, e.g. "2" for a 25 line display. Having some overlap provides a

visual reference while paging and makes scanning a document much easier.

This value is with respect to 25 screen lines. The actual value is automatically adjusted according to the size of the current window.

Config(S_TOP_MARG, "Top margin for cursor", 2)

Determines the number of lines that will normally be visible above the line the cursor is on, before the screen scrolls. Setting the value greater than "0" ensures that you will normally see some lines above your present cursor position.

This value is with respect to 25 screen lines. The actual value is automatically adjusted according to the size of the current window.

Config(S_BOT_MARG, "Bottom margin for cursor", 2)

This is similar to the previous parameter; it determines the number of lines that will normally be visible below the line the cursor is on, before the screen scrolls. Setting the value greater than "0" ensures that you will normally see some lines below your present cursor position.

Config(S_STAT_C, "Status line character", 32)

(DOS, UNIX, QNX only) The empty parts of the 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" on an IBM PC.

Config(S_CONT_C, "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 4.

Config(S_RESTORE, "Restore screen mode on exit", 2)

(IBM DOS only) When set to "1", VEDIT will upon exit restore the screen size and mode to the same as when it started up. When set to "0", it returns to DOS in the current screen mode.

Config(S_MONO, "Enable monochrome (B&W) colors", 0)

(DOS, QNX only) Same as **{CONFIG, Colors, Enable monochrome (B&W) colors}**.

Config(SR_CASE_OPT, "Default case-sensitive option", 0)

Same as **{CONFIG, Search options, Default Case-sensitive option}**.

Config(SR_MODE_OPT, "Default search mode", 1)

Same as **{CONFIG, Search options, Default search mode}**.

Config(SR_RES_POS, "Restore edit position on error (0 - 2)", 1)

Same as **{CONFIG, Search options, Restore edit position on error}**.

Config(U_E_DIALOG, "Enable full dialog boxes", 1)

(DOS, UNIX, QNX only) Same as **{CONFIG, Misc, Enable full dialog boxes}**.

Config(U_FULL_DLG_O, "Full dialog box options (0 - 31)", 31)

(DOS, UNIX, QNX only) Same as **{CONFIG, Misc, Full dialog box options}**.

Config(U_TERSE_DLG_O, "Terse dialog box options (0 - 31)", 31)

(DOS, UNIX, QNX only) Same as **{CONFIG, Misc, Terse dialog box options}**.

Config(U_E_SUBM_P, "Sub-menu preview (0=Off, 1, 2)", 1)

(DOS, UNIX, QNX only) Same as **{CONFIG, Display options, Enable sub-menu preview}**.

Config(U_KEY_IN_O, "Keyboard input options (0 - 31)", 17)

Same as **{CONFIG, Misc, Keyboard input options}**.

Config(U_BEEP_LVL, "Beep level (0=Never, 1 - 5)", 2)

Same as **{CONFIG, Misc, Beep level}**.

Config(U_HELP_LVL, "Help level", 1)

Same as **{CONFIG, Misc, Help level}**.

Config(U_BOX_DRAW, "Box drawing style (0=Prompt, 1,2,3,4)", 4)

Same as **{CONFIG, Misc, Box drawing style}**.

Config(U_CM_H_CNT, "Count for Command mode help message", 3)

(VEDIT PLUS only) This parameter determines how many times the "COMMAND:" prompt is preceded by the help message:

```
Enter "exit" to exit, "v" to edit, "h" for help
```

Experienced users may prefer a value of "0" to disable it. A value of "255" causes it to always appear.

Config(U_E_UNDO, "Enable undo in Command mode", 1)

When set to "0", the undo is disabled and reset whenever macro commands are executed. When set to "1", the undo is enabled for most commands; however, it is disabled and reset when the **Call( )** command is executed. When set to "2", the undo is enabled for all commands.

Config(U_UNDO_MAX, "Undo levels - max", 1000)

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).

Config(U_UNDO_MIN, "Undo levels - min", 100)

Determines the number of levels available on the Undo stack when a newly created edit buffer shares memory with other edit buffers or text registers.

Config(U_REP_MAX, "Maximum [REPEAT] count", 256)

Same as **{CONFIG, Misc, Maximum [REPEAT] count}**.

Config(U_SHELL, "Enable DOS shell", 1)

Determines whether the functions **{MISC, DOS Shell}** and **{MISC, Run Program}** are available in the menu system. (VEDIT PLUS: also the **System( )** command.) This parameter is usually set to "1". 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 have no knowledge of DOS (or UNIX).

Config(U_DISK_EDIT, "Enable disk sector modification", 1)

(VEDIT PLUS, IBM DOS Only) The default value "1" sets "**[x] Read-only mode**" in the Disk-open dialog box so that, by default, the disk is opened in Read-only mode. With a value of "2", the disk, by default, is opened in read-write mode. With a value of "0", the disk is always opened in Read-only mode; the user cannot disable "**[x] Read-only mode**".

Config(U_LOAD_CFG, "Auto-load config...", 3)

(DOS only) Same as **{CONFIG, Misc, Auto-load config}**.

Config(U_AUTO_CFG, "Auto-save config", 1)

Same as **{CONFIG, Auto-save config}**.

Config(U_CFG_ALL, "Config all buffers", 1)

Same as **{CONFIG, Config all buffers}**.

Config(U_SAV_ENV, "Enable edit restore", 0)

Same as **{FILE, Enable edit restore}**.

Config(U_USER_P, "User configured parameter (*)", 0)

(VEDIT PLUS only) This parameter is available for any user defined purpose; it is not used internally. For example, it could be used as an "ID" to identify particular configurations — a macro would read this "ID" to ensure that the correct configuration was being run.

Config(W_WORD_WRAP, "Enable word wrap and formatting (*)", 1)

Same as **{CONFIG, Word processing, Enable word wrap and formatting}**.

Config(W_LF_MARG, "Left margin (*) (0 - 80)", 0)

Same as **{CONFIG, Word processing, Left margin}**.

Config(W_RT_MARG, "Right margin (*) (0=Win, 1=HSM, 16-255)", 0)

Same as **{CONFIG, Word processing, Right margin}**.

Config(W_JUST_PARA, "Justify paragraphs", 0)

Same as **{CONFIG, Word processing, Justify paragraphs}**

Config(W_FORMAT_O, "Format paragraph options (0 - 7)", 0)

Same as **{CONFIG, Word processing, Format paragraph options}**

Config_String( ) Parameters

Config_String(HOME, "C:\VEDIT")

Sets the *VEDIT Home Directory*. VEDIT searches here for the ".hlp" on-line help files, ".vdm" macro files and other supplemental files.

The *VEDIT Home Directory* is normally the directory into which VEDIT was installed, e.g. **`c:\vedit`** (UNIX: **`/usr/lib/vedit`**) (QNX: **`/qnx4/vedit`**).

The Windows version automatically saves this value into the **`vedit.ini`** file. In the DOS version, changes to this value must be saved into the **`vedit.exe`** file with **CONFIG, Misc, Save into VEDIT.EXE}**.

(DOS only) You can override this setting with the "VEDPATH" environment variable. For example, to use the directory "C:\UTIL\VEDIT" you could place the following command into your AUTOEXEC.BAT file:

```
SET VEDPATH=C:\UTIL\VEDIT
```

Config_String(USER_CFG, "C:\VEDIT")

Sets the *User Config Directory*. VEDIT searches here for the **`vedit.cfg`**, **`vedit.key`**, **`startup.vdm`** and **`vedit.ini`** files.

The *VEDIT Home Directory* and *User Config Directory* are usually the same. If VEDIT was installed on a network server, they will be different.

Config_String(BACKUP, "C:\VEDIT\BACKUP")

Sets the directory into which VEDIT copies modified files for backup purposes when **{CONFIG, File handling, Backup files}** is set to "2". (The default value is "1" — it creates a ".bak" file in the same directory as the original file.)

The default directory is **`c:\vedit\backup`**, but you may want to change it to **`c:\backup`**.

See "Exiting VEDIT - Backup Files" in Chapter 4 for more information.

Config_String(PR_DEF, "PRN")

Same as **{CONFIG, Printer, Change default printer}**.

Config_String(PR_START, "")

Same as **{CONFIG, Printer, Change print job start string}**.

Config_String(PR_FINISH, "")

Same as **{CONFIG, Printer, Change print job finish string}**.

Config_String(OS_PROMPT, "PG$G")

The string parameter is used as the DOS prompt when shelled out to DOS via **{MISC, DOS shell}**. The default "PG$G" gives a prompt of "*path*>>". This is equivalent to the DOS command "SET PROMPT=PG$G". For more information see your DOS manual.

Config_String(WORD_SEP, ",;:()")

Determines which characters separate words from each other. All control characters and "space" also separate words. You could include "." (period), but then "e.g." would be treated as two words. Up to 32 characters can be specified.

Config_String(PARA_SEP, ".@!\")

Determines which characters identify lines that separate paragraphs from each other. Lines beginning with these characters are considered "format command lines" that will not be merged with adjacent line when re-formatting. Blank lines also separate paragraphs. Up to 8 characters can be specified.

Config_String(MATCH_PAREN, "[][](){}{<><")

Determines which character-pairs are matched by **{GOTO, Matching ()}**. Notice how each pair is specified using three characters. Up to 8 pairs (24 characters) can be specified.

Config_String(TOOL_MENU, "Tools")

Determines the name of the **{TOOL}** menu on the main menu bar. Is is typically "Tools", "JavaTools" or "Tutorial".

Config_String(USER_MENU, "User")

Determines the name of the **{USER}** menu on the main menu bar. The documentation assumes it is "User", but any name up to 8 characters can be specified.

Config_Tab( ) Parameter

Config_Tab(8)

Determines the tab stops. This command takes the same parameter(s) as the **{CONFIG, Tab stops}** function. A single number sets a uniform tab interval; otherwise up to 32 explicit tab stops can be specified.

Chapter 9

Messages

VEDIT displays a message to notify you of errors or special conditions. Many are simply confirmation prompts to perform an operation that may be difficult 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 PLUS macro language. In most cases, the offending macro command is displayed. To resolve these, you may need to consult the VEDIT PLUS manual or the supplier of the macros.

Abandon (quit) altered file? [Yes] [No]

This is the confirmation prompt for a macro written in the VEDIT PLUS macro language that is attempting to abandon the current file. Select **[Yes]** if you really want to abandon the file without saving any changes.

VEDIT PLUS: The command option "NOCONFIRM" or "OK" suppresses this prompt on the **Buf_Empty( )** and **Buf_Quit( )** commands.

Abandon (quit) all files? [Yes] [No]

This is the confirmation prompt after you select **[Quit-all]** in the **{FILE, Exit}** dialog box. Select **[Yes]** if you really want to abandon ***all*** modified files (buffers) and exit VEDIT.

VEDIT PLUS: This is also the confirmation prompt for the **Qall( )** command. Use the command option "NOCONFIRM" or "OK" to suppress it or use the command **Qally**.

BAD FILENAME

The specified filename does not follow the conventions. Perhaps the pathname or either part of the filename is too long. Also, DOS/Windows does not allow some characters (e.g. "+", "=", "/", "*", "$", ",") in a filename.

BAD PARAMETER

You specified an invalid command parameter.

VEDIT PLUS: Numerous commands give this error if you specified an invalid argument or a numeric argument is out of range. The actual command is displayed.

BLOCK IS TOO LARGE FOR TEXT REGISTER

The block you are attempting to copy or move to a text register or with **{BLOCK, Copy/Move to cursor}** is too large; most likely it is greater than the maximum 256 Kbytes that VEDIT can currently handle.

If the block contains normal text (i.e. it is not a binary file and does not contain Null characters), you can often perform the copy/move using the Windows clipboard. The maximum block size is about half of your physical memory; however, huge clipboard operations are often very slow.

Alternatively, you can use **{BLOCK, Write to disk}** to write the huge block to a disk file, such as "block.tmp". You can then insert the block into any file at the cursor position with **{EDIT, Insert file}**.

See the on-line help topic "Blocks - Cut and paste huge blocks" (DOS: "HUGEBLOCK") for details on how to cut and paste huge blocks of any size.

BREAK

You pressed **[CANCEL]** (**<Ctrl-\>**) or **<Ctrl-C>** while printing or while a command macro is running.

This message can also occur if a macro or menu function could not complete and VEDIT could not determine its exact cause. If it persists, you may want to exit VEDIT and restart it.

[C]ANCEL, [I]GNORE, [R]ETRY? (DOS only)

Occurs when DOS detects an error and is preceded by "READ ERROR" (disk drive door open, read error), "WRITE ERROR" (disk write error) or "PRINTER NOT READY" to indicate the type of error. Press "**C**" to cancel the operation. This is safe because it returns to VEDIT. Press "**R**" to retry the operation, such as after closing the drive door. You can press "**I**" to ignore, but this usually just leads to another error.

CANNOT CREATE WINDOW

{WINDOW, Custom split} attempted to create a window that already exists or you specified too small/large a size.

VEDIT PLUS: **Win_Split( )** attempted to split a reserved window, create a window that already exists, or create too small/large a window. **Win_Reserved( )** attempted to create a second window at the top or bottom of the screen. At most, there can be one reserved window at the top and one at the bottom.

CANNOT EDIT FILE IN THIS MODE (USE BINARY)

This error is usually caused by editing a file with very long lines in non-binary mode. Try editing the file again in binary mode by immediately setting **{CONFIG, File handling, File type}** to "64".

VEDIT PLUS: Try entering the commands "**Config(F_F_TYPE,64) BOF V**" to place VEDIT into binary mode, go to the beginning of the file and reenter Visual Mode.

CANNOT ENTER CTRL-Z

You cannot enter a Ctrl-Z character into a file when **Config(F_EOF_PROC)** is enabled (it is disabled by default), because Ctrl-Z is then treated as the "End-of-file" marker.

CANNOT FIND: *search string*

The specified search string could not be found. This is a normal message for **{SEARCH}** menu items when no more occurrences can be found. Remember that VEDIT always searches from the current cursor position; perhaps you want to restart the search from the beginning of the file.

VEDIT PLUS: Use the "NOERR" option on the applicable commands to suppress this message.

CANNOT MODIFY EXECUTING MACRO REGISTER

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: A macro cannot modify the contents of any text register that contains the currently executing macro commands. This includes any parent macro when **Call( )** commands are used. In other words, self-modifying macros are not allowed.

CANNOT NEST KEYSTROKE MACROS

You previously pressed a keystroke macro that contains the **Visual( )** command to return to Visual Mode. While in Visual Mode you pressed another keystroke macro. This is not allowed — a previous keystroke macro must finish running before you can press another one. Keystroke macros rarely need a **Visual( )** command in them. They should never have **Visual( )** at the very end; they automatically return to Visual Mode.

CANNOT OVERWRITE READ-ONLY FILE

The file you specified with **{FILE, Save as}** or **{BLOCK, Write to disk}** is a read-only file and cannot be overwritten.

CANNOT RUN COMMAND.COM, ERROR #

Windows version: Most likely, DOS cannot find its COMMAND.COM file — make sure your "COMSPEC" environment variable is set correctly. Check that you can enter DOS from the normal desktop MS-DOS icon.

DOS version: Usually indicates that there is not enough memory to shell out to DOS and run COMMAND.COM. Perhaps you can exit one or more edit buffers to free more memory space. You should have V-SWAP installed in memory before running VEDIT in order to prevent this error. This error also occurs when DOS cannot find its COMMAND.COM file — make sure your "COMSPEC" environment variable is set correctly.

Cannot undo this operation! Proceed anyway? [Yes] [No]

There is insufficient free memory to delete the (large) block of text ***and*** undo it if needed. This prompt does not occur if the deletion can be undone.

CLOSE ERROR

The output file could not be closed and therefore is not saved! This is a very unusual condition, but can occur if the disk becomes write protected, or if VEDIT was attempting to save the file on a floppy disk and you prematurely removed the disk. This can happen on a network system, if the network goes down.

Unfortunately, any text that VEDIT has already attempted to write to disk is probably lost. You can attempt to save the file to another drive with **{FILE, Save as}**.

COMMAND NOT AVAILABLE IN "-B" BROWSE MODE

Some commands are not available when you have invoked VEDIT with the "**-b**" browse-only mode or "**-r**" restricted mode options. **{FILE, Save as}** is not available in either mode; **{BLOCK, Write to disk}** is not available in restricted mode.

COMMAND REQUIRES (...):

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The specified command requires parentheses and, most likely, arguments.

DIRECTORY NOT FOUND

The directory specified as part of a filename could not be found. Perhaps you mistyped it or specified the wrong drive. This error also occurs when the configured *VEDIT Home Directory* could not be found.

DOS MEMORY OR FAT ERROR. SAVE FILES AND REBOOT

This very unlikely error indicates either that memory has been corrupted, perhaps by an incompatible Memory-Resident program, or that DOS detected an error in its "File Allocation Table" (FAT). After re-booting, run the DOS "CHKDSK" program to check the integrity of the files.

END OF BUFFER REACHED

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The **Line( )** command tried to move past the beginning or end of the file. Use the "NOERR" command option to suppress this error message.

FLOW CONTROL STATEMENT STILL OPEN

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The end of a command macro was reached before the final "**}**" of a **While**, **Do-while**, **Repeat** or **For** loop was reached. Check that the "{" and "}" are properly matching in all flow control statements. It can also be caused by unpaired "**{**" and "**}**" occurring within string arguments or comments.

FILE IS BEING EDITED

The file you are trying to save to disk with **{FILE, Save as}**, **{BLOCK, Write to disk}** or **{CONFIG, Save to disk}** is currently open for editing. Open files are locked and cannot be overwritten. Use **{FILE, Buffer switch}** to see a list of files being edited.

FILE IS ALREADY OPEN IN THIS BUFFER

A macro written in the VEDIT PLUS macro language attempted to open two files in one edit buffer.

VEDIT PLUS: The **File_Open_Write( )** command attempted to open a file for output in an edit buffer that already has an output file open. Perhaps you want to change the output filename with **File_Save_As( )**.

FILE NOT FOUND

The file you specified for editing does not exist. Perhaps you mistyped the pathname or specified the wrong drive.

FILE NOT OPENED

This message follows another message and reminds you that your attempted file-open operation was cancelled. It generally follows an operating system error message if you attempt to open a file which is in use ("locked") by another program or user.

FUNCTION NOT AVAILABLE IN "BINARY" MODE

Word processing functions such as indenting and paragraph formatting cannot be performed on binary/data files, i.e. when **{CONFIG, File handling, File type}** is set to "8" or more. This prevents binary/data files from being corrupted.

IN BROWSE MODE -OR- FILE IS READ-ONLY

You cannot alter a file which is in Browse mode. A file is in browse mode if:

- It was opened with "**[x] Read-only mode**" in the file-open dialog box.
- The file has the "read-only" attribute set.
- The file is in a network directory that has been set to "read-only".
- The file is on a write-protected floppy disk.
- **{FILE, Browse mode}** has been set.
- VEDIT was invoked with the "**-b**" browse-only mode option.

INCOMPLETE COMMAND:

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The following command is missing arguments or the final ")". This error can result if the final quote of a string argument is missing.

INTERNAL ERROR # nn

VEDIT has detected an internal problem. After you press any key, VEDIT will automatically select **{FILE, Exit}**. You can then save or abandon your files. Or you can select **[Cancel]** to return to your editing; however the error will most likely immediately reoccur. After exiting, double check that your files are intact. Please contact us if you can replicate the error.

VEDIT PLUS: Following the error, VEDIT PLUS will enter Command Mode. We suggest using **Exit( )** to save your files and exit.

INVALID COMMAND:

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The specified command is not a known command. Perhaps you mistyped it or used the wrong abbreviation.

INVALID DRIVE

You specified an invalid or non-existent drive in a filename. Most likely you mistyped it.

INVALID EDIT BUFFER OPERATION

You are attempting an operation that is valid for text registers, but is invalid for edit buffers. You cannot change the contents of an edit buffer except when it is the active buffer. Copying a block to an edit buffer with **{BLOCK, Copy to register}** is not allowed. This error also occurs if you attempt to execute the contents of the current buffer as a command macro with **{MISC, Execute macro}** — you must first switch to another buffer.

VEDIT PLUS: May also be caused by an improper **Reg_Copy( )**, **Reg_Empty( )** or **Reg_Load( )** command.

INVALID EXPRESSION:

The numeric expression you entered at the prompt for a number has incorrect syntax.

VEDIT PLUS: The numeric expression used as a numeric argument has incorrect syntax.

INVALID FLOW CONTROL

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: A **While**, **Do-while**, **Repeat**, **For** or **If** statement has incorrect syntax. Perhaps you left off the condition or the initial "{".

INVALID [HELP] REQUEST

During the **[HELP]** function you are prompted to press the function/control key for the desired edit function. Instead, you pressed a displayable character or an unused function/control key.

INVALID KEY 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 MENU

The custom **{USER}** menu that you attempted to create contains a syntax error. The **{USER}** menu is typically loaded with the command **Reg_Load(124,"user.mnu")** in the **`startup.vdm`** file. In this case the "user.mnu" file has a syntax error. See "{USER} and {TOOL} Menus" in Chapter 5.

INVALID REGISTER

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: You specified an invalid numeric register for a numeric command. The numeric register must be in the range "0" - "127". Or you specified an invalid text register number which must be in the range "0" - "127".

INVALID TEXT MARKER

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: You specified an invalid text marker with the **Set_Marker( )** or **Marker( )** command. The text markers must be in the range "0" - "9".

JUMPING INTO A FLOW CONTROL STATEMENT

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The **Goto** command cannot jump into a flow control statement. Flow control statements can only be entered at their beginning. However, it is allowable to jump out of a flow control statement, or to jump within it.

KEYBOARD LAYOUT CORRUPTED

VEDIT noticed that its internal keyboard layout table has become corrupted; it will immediately exit after saving all files. May be caused by loading an invalid .KEY file with **{CONFIG, Keyboard layout, Load layout}**. If this error occurs whenever you start VEDIT, simply delete the file **`vedit.key`** file in the *VEDIT Home Directory*. VEDIT will then start up with the "normal" layout. Then select **{CONFIG, Keyboard layout, Save layout}** to create a new **`vedit.key`** file.

DOS version: You may have to go back to the original VEDIT.EXE file if the invalid keyboard layout was saved into the current VEDIT.EXE file.

LABEL MISSING: *label***:**

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: The **Goto** command could not find the "*label*:" it is to jump to. Perhaps the ":" is missing.

MACRO ERROR IN *r*

This message often precedes other error messages to indicate in which text register the offending command occurred. '*r*' has value 255 for command macro errors occurring in keystroke macros.

NESTING (STACK) ERROR

Caused by a programming error within a VEDIT PLUS command macro.

VEDIT PLUS: You cannot nest command macros deeper than 25 levels. This error often is caused by a macro that recursively calls itself — e.g. register 10 contains the command **Call(11)** and register 11 contains the command **Call(10)**. Flow control structures cannot be nested deeper than 25 levels. Using commands as arguments to other commands cannot be nested deeper than five levels. The **Save_Pos()** command can only save five edit positions on its stack. Also caused when **Reg_Push( )** or **Num_Push( )** attempt to push too many registers. Also caused by **Out_Ins( )** that doesn't have a matching **Out_Ins(CLEAR)**; similarly for other re-direction commands.

New file

This message briefly displays on the status line if the file opened for editing did not exist and a new file has been created. If you typed the wrong filename, you can edit the correct file by immediately selecting **{FILE, Open, Same buffer}** (**<Alt-O>**).

No assigned filename! Enter "Save As" filename:

There is no filename assigned into which to save the current edit buffer, e.g. with **{FILE, Exit}**. VEDIT is prompting you for the desired filename.

NO CURRENT SEARCH/REPLACE STRING

The **{SEARCH, Again}** function did not operate because you have not yet specified a search or replace string with **{SEARCH, Search}** or **{SEARCH, Replace}**.

NO DIRECTORY SPACE

NO DISK SPACE

The disk (or directory) became full before the entire file(s) was saved to disk. To save your file(s) you must first delete some unneeded files. Alternatively, you might be able to save the file to another disk using **{FILE, Save as}**.

Windows version: Switch to Explorer (or Program Manager) and delete any old unneeded files to make more free disk space.)

DOS version: Shell to DOS with **{MISC, DOS Shell}** and delete any old unneeded files.

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 that have "$" in their filename extension, e.g. ".1$$" and ".1R$". They are temporary files needed by VEDIT. (You can delete them after exiting VEDIT in the unlikely event they still exist.)

NO EDIT BUFFER AVAILABLE

You are attempting to simultaneously edit more files than VEDIT has available edit buffers. At most 32 files can be simultaneously edited; however command macros can use edit buffers for their own purposes and reduce this number. Perhaps you can close some of the files you are done editing with **{FILE, Close}**.

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, such as copying too large a block to a text register, this error will rarely occur. See the error message "BLOCK IS TOO LARGE" for a method of copying larger blocks.

This error will occur with **{EDIT, Insert file}** if the current buffer does not yet have a file open. First open a file with **{FILE, Open}** or **{FILE, Save as}**.

Be sure that configuration parameter **Config(F_AUTO_BUF)** is set to "2" as it always should be. (Other settings are provided for backwards compatibility with old macros.)

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 close other files.

If there is sufficient free memory, try editing the file in binary mode by setting **{CONFIG, File handling, File type}** to "64".

NOT ENOUGH MEMORY TO LOAD VEDIT (DOS Only)

There is insufficient memory available for VEDIT to start up — approximately 180K is needed. Use the DOS "MEM" or "CHKDSK" command to see how much is available.

NOT FOUND IN HELP FILE

The help topic you selected could not be found in the on-line help file. Most likely you entered the topic name incorrectly.

Ok to truncate (erase) output file? [Yes] [No]

VEDIT users should never see this error message.

VEDIT PLUS: This is the confirmation prompt for the **File_Truncate( )** command which truncates and closes the current output file. (The word "erase" is to remind you that improper use of this command can erase the file being edited!) Use the command option "NOCONFIRM" or "OK" to suppress the confirmation.

Ok to erase these files? [Yes] [No]

This message only occurs when a macro written in the VEDIT PLUS macro language uses the **File_Delete( )** command to delete files from disk. Select **[Yes]** to delete the listed files.

Ok to overwrite existing file? [Yes] [No]

The functions **{BLOCK, Write to disk}** and **{CONFIG, Keyboard layout, Save layout}** ask for confirmation if the designated file to be written already exists on disk. Select **[Yes]** to overwrite the existing disk. Note that these functions ***DO NOT*** create a "backup" of any existing file.

VEDIT PLUS: The **Reg_Save( )**, **Config_Save( )** and **Write_Block( )** command asks for confirmation if the designated file to be written already exists on disk. Use the command option "NOCONFIRM" or "OK" to suppress the confirmation.

PRINTER/DEVICE NOT READY (DOS Only)

The printer does not respond — most likely it is not turned on or is not set "on-line". Under DOS, this message is followed by "[C]ANCEL, [I]GNORE, [R]ETRY". Press "**R**" to retry after setting your printer on-line. 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. Because many printers have "print buffers", your printer may continue printing for some time after you press **<CTRL-C>**.

READ ERROR

An error occurred reading from disk — perhaps the drive door is not closed or you are trying to read an unformatted floppy disk. Under DOS, this message is followed by "[C]ANCEL, [I]GNORE, [R]ETRY". Press "**R**" to retry after closing the drive door. Press "**C**" to cancel the read command. If the disk has developed a bad sector, you can press "**I**" to ignore the error, but this will likely read a block of garbage from the disk.

Redefine displayable char? [Yes] [No]

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. Select **[Yes]** to redefine it, or **[No]** if you made a mistake and do not want to redefine it.

Redefine existing key? [Yes] [No]

The "Function/Control Key" you entered while adding a new keystroke macro is already assigned to an edit function or a keystroke macro. Select **[Yes]** to redefine it, or **[No]** if you made a mistake and you do not want to redefine it.

VEDIT PLUS: The command option "NOCONFIRM" or "OK" suppresses this prompt on the **Key_Add( )** command.

REGISTER NOT AVAILABLE / PROTECTED

Command macros can protect the text registers they use internally so that they are not accidentally modified from Visual Mode. We suggest that text registers 0 through 9 be used as Visual Mode "cut and paste" registers. Command macros should only use registers "10" and up for their own use. Ideally, command macros should use the highest numbered registers possible to reduce the chance of this error.

REGULAR EXPRESSION SYNTAX ERROR (SEARCH)

There is a syntax error on the search side of a regular expression. Perhaps a "]" or "}" 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.

SEARCH / REPLACE STRING TOO LONG

This error occurs if the overall search/replace string, plus any "variable" characters included with "**|@(*r*)**", exceeds 260 characters.

SYNTAX ERROR (SEARCH)

There is a syntax error in the search string using pattern matching codes. The search string cannot end in just "|"; use "||" to search for a "|". "|*ddd*" requires that all three decimal digits are present; e.g. use "**|000**" to search for the "Null" character. "|H*hh*" requires that both hexadecimal digits are present; e.g. use "**|H0D**" to search for the Carriage-Return character.

TOO MANY FILES OPEN

You are attempting to simultaneously edit more files than the "FILES=nn" statement in your CONFIG.SYS file allows. Increase the number by three for each additional file you want to edit. See "Checking your CONFIG.SYS file" in Chapter 2 (Getting Started).

V-SWAP ERROR #n (DOS Only)

VEDIT attempted to use V-SWAP while shelling out to DOS and an error occurred. See "V-SWAP Error Messages" in the on-line help topic "VSWAP" for a detailed description.

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 — perhaps the drive door is not closed or you are trying to write to an unformatted floppy disk. Under DOS this message is followed by "[C]ANCEL, [I]GNORE, [R]ETRY". Press "**R**" to retry after closing the drive door. Press "**C**" to cancel the write command. If your disk has developed a bad sector, you should attempt to save the file to another disk using **{FILE, Save as}**.

Appendices

A - File Management

This is a technical description of VEDIT's 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 read from disk (opened), modified, and then written back to disk (closed) with either the original filename or a new filename.

Since VEDIT can edit files that 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; for smaller files, all of it is read into memory.
2. A new file is created for the "output file". It temporarily has the desired filename, but with an extension of "*.r*$$", where '*r*' is the edit buffer number. The main buffer #1 uses the extension ".1$$". Buffer #10 uses the extension ".A$$", buffer #11 uses ".B$$" and so on.
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 performs some file renaming — it first renames the existing input file to have an extension of ".BAK". This is referred to as the "*backup*" of the file. Any previous backup of the file is deleted by this process.

   It then renames the "*.r*$$" output file to the real name of the destination file.

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*".

Conceptually, 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 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*R$" where '*r*' is the edit buffer number.

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 that can be edited is 1/2 of a disk. (Reading from the input file does not free up disk space.) If the input and output files are on different drives, the maximum file size is a full disk. Due to the additional temporary file needed for backward file buffering, the maximum file size (in the worst case) is reduced to 1/3 of a disk. The temporary file is always on the current drive. (With a three drive system you could safely edit a file one disk in length, by making the current, the input file and output file drives all different.) These file size limitations arise because in the worst case VEDIT needs to create a

temporary file which is nearly as large as the output file, which is generally as large as the input file.

It is always best be to sure that there is enough free disk space before editing a file. The DOS "DIR" command can tell you the size of the file being edited and the amount of free disk space.

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 ".BAK" 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.

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, Open, Same buffer}** (**<Alt-N>**) to start editing the file over again. This has the added benefit of saving the current file.

Networking and Multi-Tasking

Some operating systems, such as UNIX and QNX allow several programs to be run simultaneously on one computer system by one or more users. Networks also allow multiple users to access a common set of files. These environments must deal with the situation where one program attempts to access a file which is already in use by another program. In effect, the second program is denied access to the file, or "locked out". This process is called "*file locking*". For example, two users cannot simultaneously run VEDIT on the same file.

VEDIT is designed to work in both environments and in conjunction with their file locking. Typically, if you try to access a file with VEDIT which is already in use by another program, the operating system will first issue you an error message. Then VEDIT will issue an additional error message "FILE NOT OPENED" to note that the file was not successfully accessed. VEDIT ensures that files which it is working on, or will soon need to access, are locked from use by other programs. VEDIT will also release files as soon as it is done with them so that they may then be used by other programs.

VEDIT also prevents 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.

B - Search Modes Summary

Pattern Matching Codes

NOTE: Only the codes "**|H*hh***", "**|N**", "**|O*ooo***", "**|*ddd***" and "**|@(*r*)**" can be used on the replacement side.

|A	Match any alphabetic letter, upper or lower case.
|B	Match a blank - one space or tab.
|C	Match any control character.
|D	Match any numeric digit - "0" - "9".
|F	Match any alphanumeric - a letter or a digit.
|G	Match any graphics (high-bit) character.
|H*hh*	Match the character with hexadecimal value '*hh*'. Can also be used on the replacement side.
|I	Match any word separator, including **Config_String(WORD_SEP)**.
|K	Match any non-standard control character other than Tab, Carriage-Return and Line-Feed.
|L	Match "newline": Carriage-Return and/or Line-Feed. CR is optional in DOS/Windows files.
|M	Multi - match any sequence of zero or more characters.
|N	Match "newline" characters, similar to "**|L**". CR is required in DOS/Windows files. Can also be used on the replacement side.
|O*ooo*	Match the character with octal value '*ooo*'. Can also be used on the replacement side.
|P	Match any "parenthesis" - { } [ ] ( ) < >.
|S	Match any separator - not a letter, digit or "_" (underscore).
|T	Match the Tab character (value 09).
|U	Match any upper case letter.
|V	Match any lower case letter.
|W	Match white space - single or multiple Spaces or Tabs.
|X	Match extended white space - one or more Spaces, Tabs, Carriage-Returns and/or Line-Feeds.
|Y	Match multiple characters until the next pattern matches.
|*ddd*	Match the character with decimal value '*ddd*'.
|000	Match the Null (value 00) character
|<	Match beginning of line (zero length match).
|>	Match end of line (zero length match).
|{*set*}	Matches one occurrence of any item in the "pattern set".
|[*set*]	Matches one optional occurrence of any item in "pattern set".
|@(*r*)	Access contents of text register '*r*' as a variable string.
|?	Match any character.
|!	Match any character except following character or pattern.
| |	Use "| |" when you need to search for a "|".

Regular Expressions

Expressions that match a single character:

.	(Period) Simple wildcard that matches any character.
[*list*]	Matches any one character in the '*list*'.
[^*list*]	Matches any one character not in the '*list*'.
[~*list*]	Same. "**[~**" is equivalent to "**[^**".
\b	Matches the ASCII backspace character (hex 08).
\d*DDD*	Matches the character with decimal value '*DDD*'. All three digits ***MUST*** be present.
\e	Matches the ASCII <Esc> character (hex 1B).
\f	Matches the ASCII Form-feed character (hex 0C).
\h*HH*	Matches the character with hexadecimal value '*HH*'. Both digits ***MUST*** be present.
\n	Matches the Line-Feed character (hex 0A). This is the "newline" character for UNIX type text files. To search for multiple-line patterns, use "**\N**" instead.
\N	Matches the "newline" character(s) and allows searching for multiple line patterns. The "newline" depends upon the current file type and can be CR, CR+LF or LF. ("**\N+**" and "**\N***" are currently not supported.)
\o*OOO*	Matches the character with octal value '*OOO*'. All three digits ***MUST*** be present.
\r	(Lower case) Matches the ASCII CR character (hex 0D).
\s	Matches the ASCII space character (hex 20).
\t	Matches the ASCII tab character (hex 09).
\0	(Zero) Matches the ASCII Null character (hex 00).
****	"\" followed by a special character matches that character. The special characters are: ^ $. * + ? - ~ \ | [ ] { }

Expressions that match multiple characters:

*****	Matches zero or more occurrences of the ***preceding*** single character matching expression.
+	Matches one or more occurrences of the ***preceding*** single character matching expression.
?	Matches zero or one occurrences of the ***preceding*** single character matching expression.
\1 - \9	Matches the same text as was matched by the previous '*n*'th group.

Other:

^	(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).
{ }	Groups expressions for future reference in either the search string or replacement string.

|	Matches any text that is matched by the preceding OR the following expression. It cannot occur within { }.
\@(*r*)	Use the contents of text register '*r*' in this position in the search (or replace) string.

Replacement Side:

\b	The ASCII backspace character (hex 08).
\d*DDD*	The character with decimal value '*DDD*'. All three digits ***MUST*** be present.
\e	The ASCII <Esc> character (hex 1B).
\f	The ASCII Form-feed character (hex 0C).
\h*HH*	The character with hexadecimal value '*HH*'. Both digits ***MUST*** be present.
\n	The Line-Feed character (hex 0A). This is the "newline" character for UNIX type text files.
\N	The "newline" character(s) depending upon the current file type and can be <CR><LF>, <LF> or <CR>.
\o*OOO*	The character with octal value '*OOO*'. All three digits ***MUST*** be present.
\r	The Carriage-Return character (hex 0D).
\s	The ASCII space character (hex 20).
\t	The ASCII tab character (hex 09).
\0	(Zero) The ASCII Null character (hex 00).
\@(*r*)	Use the contents of text register '*r*' in this position in the replacement string.
\1 - \9	Same text as was matched by the n'th group on the search side.
&	Entire text that was matched by the search expression.

Precedence of Regular Expression Operators:

Regular Expression Operator Precedence

Highest:	\
	[ ]
	* + ?
	{ }
	Concatenation
Lowest:	|

C - Application Notes

This appendix lists a few keystroke macros that you can add to VEDIT. These and many more are listed in the supplied file KEY-MAC.LIB. Some of these macro are included in the default **{USER}** menu. The VEDIT PLUS macro language is used in these keystroke macros.

NOTE: The Chapter 4 topic "Editing the Keyboard Layout - Adding a Keystroke Macro from KEY-MAC.LIB" describes how to add these keystroke macros to VEDIT.

Duplicate line

This keystroke macro duplicates the current line of text and moves the cursor to the beginning of the new line.

```
[VISUAL EXIT] BOL( ) Block_Copy( )
```

This simple macro above does not work quite right on the last line of a file. However, the following macro does.

```
[VISUAL EXIT]
BOL( ) Block_Copy( ) if (Cur_Col>1) { Ins_Newline( ) }
```

Move by sentence

The following keystroke macro moves the cursor to the beginning of the next sentence.

```
[VISUAL EXIT] Search(".|S") Search("|F")
```

Transpose letters

The following keystroke macro transposes two characters and advances the cursor to the following character.

```
[VISUAL EXIT] Block_Move(Cur_Pos+1,Cur_Pos+2) Char( )
```

The following variation transposes the previous two characters without advancing the cursor.

```
[VISUAL EXIT] Block_Move(Cur_Pos-2,Cur_Pos-1)
```

Insert date and time

This keystroke macro inserts the current date and time at the cursor position.

```
[VISUAL EXIT]
Out_Ins( ) Date(NOCR) Type_Space(2) Time(NOCR)
Out_Ins(CLEAR)
```

Scroll to center the current line

This keystroke macro scrolls the screen so that the current line is centered vertically in the middle of the screen.

```
[VISUAL EXIT] Set_Visual_Line(Win_Lines/2)
```

Simultaneously [PAGE UP/DOWN] two windows

These keystroke macros perform a simultaneous **[PAGE UP]** and **[PAGE DOWN]** in two windows. (It beeps if there are not exactly two windows open.) Assign them to any two available keys, perhaps <**Alt-PgUp**> and <**Alt-PgDn**>.

```
[VISUAL EXIT]
if (Win_Total==2) { #100=Win_Num
Win_Switch(Win_Next,ATTACH)
Do_Visual("\PU\") Update() Win_Switch(#100,ATTACH)
Do_Visual("\PU\") } else { Alert() }
```

```
[VISUAL EXIT]
if (Win_Total==2) { #100=Win_Num
Win_Switch(Win_Next,ATTACH)
Do_Visual("\PD\") Update() Win_Switch(#100,ATTACH)
Do_Visual("\PD\") } else { Alert() }
```

Copy block to another open file (buffer)

This keystroke macro copies the highlighted block of text to another open file's (buffer's) cursor position. If three or more files are being edited, it prompts for the buffer number.

```
[VISUAL EXIT]
if (be == -1){ return } Num_Push(10,10) #10=Buf_Num
Reg_Copy_Block(0,bb,be,RESET) if (bt == 2){
Buf_Switch(Buf_Next) } else {
Buf_Switch( Get_Num('Copy to which buffer? ',STATLINE)) }
Reg_Ins(0) Buf_Switch(#10) Num_Pop(10,10)
```

Search and list matching lines

This macro prompts for a text string and lists all lines containing the string with their line numbers. See also **{USER, Search and select}**.

```
[VISUAL EXIT]
Save_Pos()
Get_Input(103,"Enter text string: ",STATLINE+NOCR)
Win_Clear() Begin_Of_File()
while (Search(@103,ADVANCE+NOERR)) {
Num_Type(Cur_Line,NOCR) Type_Space(2)
Type(0) Type() }
Restore_Pos()
```

Delete blank lines

This keystroke macro deletes all blank lines, i.e. lines containing only a Carriage-Return and/or Line-Feed. Does not delete “blank” lines that also contain spaces and tabs.

```
[VISUAL EXIT]
Replace(“|<|>|N”,"",BEGIN+ALL+NOERR)
```

The following macro also deletes “blank” lines that contains spaces and tabs.

```
[VISUAL EXIT]
Replace(“|<|[|W]|>|N”,"",BEGIN+ALL+NOERR)
```

Delete lines containing a particular string

This keystroke macro prompts for a search string and then deletes all lines in the file containing the string. As a precaution, the search string must be at least four characters long; however the macros should still be used with CAUTION! The cursor is left following the last deleted line.

```
[VISUAL EXIT]
Get_Input(103,"Enter search string: “,STATLINE+NOCR)
if (Reg_Size(103) < 4) {
   Alert()
   Statline_Message(”ERROR - Minimum 4 chars") Return
}
BOF()
repeat(ALL) {
   Search(“|@(103)”,ERRBREAK)
   BOL() Del_Line()
}
```

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. It is the same as **{USER, Word count}**.

```
[VISUAL EXIT]
M(“Counting words in file. Please wait...”,STATLINE)
Save_Pos( )
BOF( ) #102=Search(“|s|a”,ALL+NOERR)
Restore_Pos( )
#101=Win_Num
Win_Switch(STATLINE)
Win_Clear( )
M(“Word count = ”) Num_Type(#102,NOCR)
Get_Key(“   Press any key...”)
Win_Switch(#101)
```

Compare two windows

This keystroke macro compares the current window with the "next" window. It is similar to **{Search, Compare buffers}**, but it never prompts for the buffer number. It beeps if there is only one window.

```
[VISUAL EXIT]
if ((#100=Win_Next)!=Win_Num) {
   Compare(Win_Status(#100)+BUFFER)
} else { Alert() }
```

Align left edge of line with cursor

This keystroke macro aligns the left edge of the current line with the position of the cursor and advances to the next line. Repeatedly pressing the assigned key aligns the following lines. This is useful for re-aligning text in a C program.

```
[VISUAL EXIT]
#100=Win_Hor-1 BOL() Search("|!|W") Del_Line(0)
Ins_Text(" ",COUNT,#100) Do_Visual("\CD\")
```

Running V-SPELL from within VEDIT

This keystroke macros runs the V-SPELL spelling corrector from within VEDIT to correct the current file being edited. You can assign it to **<Ctrl-F1>** or any other desired key.

Note: The **Out_Reg( )** and **Dir( )** commands are needed to convert long filenames to the short 8.3 format needed by V-SPELL.

```
[VISUAL EXIT]
Out_Reg(102) Dir(pathname,NOMSG+SHORT)
Out_Reg(CLEAR) File_Close(NOMSG)
Sys("vs |@(102)",DOS+MAX+DELETE+NOMSG)
File_Open(@102)
```

Toggle into VGA 132 column mode (DOS Only)

This capability is not directly built into VEDIT because it is very non-standard and different for each VGA card.

This keystroke macro switches some VGA cards into 132 x 25 mode using video mode 35 (23 hex). Substitute "35" for other modes available on your particular VGA card.

```
[VISUAL EXIT] #100=35 Sys_Int(16) Screen_Reset( )
```

You can return to 25 line mode with **{MISC, VGA/EGA toggle}** or with the following keystroke macro.

```
[VISUAL EXIT] #100=3 Sys_Int(16) Screen_Reset( )
```

NOTE: Video hardware manufactures warn that incompatible video modes may damage the video card and/or monitor. Therefore, only select video modes that are supported by your hardware.

D - Troubleshooting (DOS)

DOS:
WINDOWS:

Refer to the topic "Configuration - Troubleshooting" in Chapter 8 if you are having trouble saving configuration changes.

Hangs immediately

If VEDIT hangs immediately or after a few keystrokes, remove any keyboard related Terminate-stay-resident programs and try it again. The programs are often loaded by your AUTOEXEC.BAT file.

In some rare cases VEDIT may crash at startup if **{CONFIG, Misc, Keyboard polling}** or **{CONFIG, Misc, Keyboard repeat rate}** are set to any value other than "0". If this occurs, start up VEDIT with the invocation options "**-j**" and "**-k**" which force these settings to "0" (their most compatible values).

vedit -j -k

Then select **{CONFIG, Misc, Save into VEDIT.EXE}** to make these settings permanent.

See the directions below for "Keyboard typematic rate...".

Keyboard appears "hung" after exiting VEDIT

Follow the directions below for "Keyboard typematic rate...".

Try tapping the **<Alt>** and **<Ctrl>** keys to see if this clears the "hung" condition. For reasons unknown to us, some systems temporarily get the **<Alt>** and **<Ctrl>** keys "stuck" in the down position when they really are not. Users reporting this problem report the same problem with other programs.

Cannot read menus on monochrome display

Most likely VEDIT was installed for a color display and you have a Laptop computer or a monochrome display connected to color adapter. Start up VEDIT with the "**-m**" option and then select **{CONFIG, Save into VEDIT}**.

Cannot read first letter in menus

This occurs with some monochrome displays and Laptops. First adjust your screen contrast and brightness controls to see if this helps. If not, select the **{CONFIG, Colors}** sub-menu and change "Color for menu bar selection" and "Color for menu selection" to "7", "15" or "127". Experiment with these and other values.

Keyboard typematic rate is slow in VEDIT or after exiting

VEDIT normally speeds up the keyboard typematic rate (key repeat rate) inside VEDIT and then restore it upon exit. However, this interferes with other keyboard enhancers and many PS/2 machines. In this case configure **{CONFIG, Misc, Keyboard typematic rate}** to "0". With this setting,

VEDIT does not attempt to change the typematic rate. Then select **{CONFIG, Save into VEDIT}**. Finally, exit and start up VEDIT again.

For more details see the topic "Installation - Setting the Keyboard Typematic Rate" in Chapter 2 (Getting Started).

Mouse does not work or works erratically in VEDIT

For VEDIT's mouse support to work you must have a mouse driver installed, typically this is the line "DEVICE=MOUSE.SYS" in your CONFIG.SYS file or the line "MOUSE" in your AUTOEXEC.BAT file. If the mouse cursor does not move to the lower lines of a 43 or 50 line display, you should check that you have a recent version of the mouse driver, version 6.24 or later for the Microsoft mouse, version 4.15 or later for the Logitech mouse.

The mouse support is optimized for the Microsoft and Logitech mouse; if you have a Mouse Systems mouse, you may have to set **{CONFIG, Misc, Mouse options}** to "65" or "66". Then select **{CONFIG, Save into VEDIT}**, exit and start up VEDIT again.

If your mouse still acts erratically, you can try setting **{CONFIG, Misc, Mouse cursor}** to "129" or "130". Then select **{CONFIG, Save into VEDIT}**, exit and start up VEDIT again. These special values allow VEDIT to work with older style mouse drivers.

If you still have trouble with the mouse, select the **{CONFIG, Misc}** sub-menu and press **[HELP]** (**<F1>**). Additional values for "Mouse options" that solve other mouse problems are described.

With compatible VGA cards and mouse drivers, VEDIT's mouse support will work with extended screen sizes such as 132 by 50. If your mouse does not work in these extended modes, you should contact the VGA card and/or mouse manufactures for a compatible mouse driver.

After exiting VEDIT, DOS only scrolls in the top half of the 43/50 lines on the screen.

When running VEDIT on an EGA/VGA system, you can toggle to 43/50 line mode. If you then exit VEDIT and configuration parameter **Config(S_RESTORE)** is set to "0", your screen will stay in the 43/50 line mode and DOS will normally run properly with the additional lines. However, if you notice that DOS is only running in the top 25 lines, the most likely cause is having ANSI.SYS installed. ANSI.SYS does not support more than 25 lines. Many EGA/VGA cards come with a utility disk containing an improved ANSI.SYS which supports 43/50 lines.

E - IBM PC Keyboard Layout

The entire "normal" keyboard layout is listed in alphabetic order by key name. It includes the actual contents of each keystroke macro. This list is similar to that displayed by **{CONFIG, Keyboard layout, Edit/view layout}**.

<Alt-A>	[MENU] CKA	
<Alt-B>	[MENU] B	
<Alt-C>	[MENU] C	
<Alt-D>	[MENU] VD	
<Alt-E>	[MENU] E	
<Alt-F>	[MENU] F	
<Alt-G>	[MENU] G	
<Alt-H>	[MENU] H	
<Alt-I>	[MENU] BN	
<Alt-J>	[MENU] VC	
<Alt-K>	[MENU] CKR	
<Alt-L>	[MENU] BL	
<Alt-M>	[MENU] M	
<Alt-N>	Used by Compiler support	
<Alt-O>	[MENU] FMS	
<Alt-P>	Used by Compiler support	
<Alt-Q>	[MENU] VI	
<Alt-R>	[REPEAT LAST]	
<Alt-S>	[MENU] S	
<Alt-T>	[MENU] T	
<Alt-U>	[MENU] U	
<Alt-V>	[MENU] V	
<Alt-W>	[MENU] W	
<Alt-X>	[MENU] FX	
<Alt-Y>	[MENU] FMH	
<Alt-Z>	[MENU] VZ	
<Alt-`>	Not assigned	
<Alt—>	Used by Windows	
<Alt-=>	[MENU] VB	
<Alt-\>	[MENU] VH	
<Alt-[>	[MENU] GN	
<Alt-]>	[MENU] GD	
<Alt-;>	Not assigned	
<Alt-'>	Not assigned	
<Alt-,>	Not assigned	
<Alt-.>	[MENU] VV	(DOS Only)
<Alt-/>	[ESCAPE] W	

<Alt-Bksp>	[MENU] EUE
<Alt-Cursor Down>	[SCROLL DOWN]
<Alt-Cursor Left>	[SCROLL LEFT]
<Alt-Cursor Right>	[SCROLL RIGHT]
<Alt-Cursor Up>	[SCROLL UP]
<Alt-Del>	Not assigned
<Alt-End>	Not assigned
<Alt-Enter>	[MENU] HD
<Alt-F1>	[MENU] HK
<Alt-F2>	[MENU] SR
<Alt-F3>	Not assigned
<Alt-F4>	[MENU] FX
<Alt-F5>	[MENU] WS
<Alt-F6>	Cannot be used; reserved by Windows
<Alt-F7>	Not assigned
<Alt-F8>	Not assigned
<Alt-F9>	[MENU] BM
<Alt-F10>	[VISUAL ESCAPE]
<Alt-F11>	[T-REG MOVE]
<Alt-F12>	[MENU] FS
<Alt-Home>	Not assigned
<Alt-Ins>	Not assigned
<Alt-PgDn>	Not assigned
<Alt-PgUp>	Not assigned
<Backspace>	[BACKSPACE]
<Ctrl-A>	[MENU] BA
<Ctrl-B>	[MENU] EF
<Ctrl-C>	[MENU] EC
<Ctrl-D>	[MENU] GS
<Ctrl-E>	[VISUAL EXIT]
<Ctrl-F>	[MENU] SS
<Ctrl-G>	[MENU] GG
<Ctrl-H>	[MENU] SR
<Ctrl-I>	[MENU] SI
<Ctrl-J>	[ERASE BOL]
<Ctrl-K>	[ERASE EOL]
<Ctrl-L>	[ERASE LINE]
<Ctrl-M>	Not assigned
<Ctrl-N>	[MENU] FN
<Ctrl-O>	[MENU] FO
<Ctrl-P>	[MENU] FP
<Ctrl-Q>	[ENTER CTRL]
<Ctrl-R>	[REPEAT]

<Ctrl-S>	Not assigned
<Ctrl-T>	Not assigned
<Ctrl-U>	[MENU] EUL
<Ctrl-V>	[MENU] EP
<Ctrl-W>	[MENU] WO
<Ctrl-X>	[MENU] ET
<Ctrl-Y>	[MENU] EUR
<Ctrl-Z>	[MENU] EUE
<Ctrl-[>	Not assigned
<Ctrl-\>	[CANCEL]
<Ctrl-]>	[MENU] GM
<Ctrl-Shift-^>	[ENTER CTRL]
<Ctrl-_>	Not assigned
<Ctrl-Bksp>	[DEL PREV WORD]
<Ctrl-Cursor Down>	[NEXT PARAGRAPH]
<Ctrl-Cursor Left>	[PREV WORD]
<Ctrl-Cursor Right>	[NEXT WORD]
<Ctrl-Cursor Up>	[PREV PARAGRAPH]
<Ctrl-Del>	[DEL NEXT WORD]
<Ctrl-End>	[MENU] GE
<Ctrl-Enter>	[NEXT LINE]
<Ctrl-F1>	[MENU] HS
<Ctrl-F2>	[MENU] SA
<Ctrl-F3>	[MENU] SC
<Ctrl-F4>	[MENU] WO
<Ctrl-F5>	[MENU] WP
<Ctrl-F6>	[MENU] WN
<Ctrl-F7>	[MENU] ML
<Ctrl-F8>	[MENU] ME
<Ctrl-F9>	[MENU] BC
<Ctrl-F10>	[VISUAL EXIT]
<Ctrl-F11>	[T-REG COPY]
<Ctrl-F12>	Not assigned
<Ctrl-Home>	[MENU} GB
<Ctrl-Ins>	[MENU] EC
<Ctrl-PgDn>	[SCREEN END]
<Ctrl-PgUp>	[SCREEN BEGIN]
<Ctrl-Shift-C>	[MENU] ESC
<Ctrl-Shift-O>	[MENU] FMQ
<Ctrl-Shift-P>	[MENU] MP
<Ctrl-Shift-R>	[MENU] MQ
<Ctrl-Shift-V>	[MENU] ESP
<Ctrl-Shift-X>	[MENU] EST

<Ctrl-Shift-Z>	[MENU] EUL
<Ctrl-Tab>	[MENU] WN
<Cursor Down>	[CURSOR DOWN]
<Cursor Left>	[CURSOR LEFT]
<Cursor Right>	[CURSOR RIGHT]
<Cursor Up>	[CURSOR UP]
<Del>	[DELETE]
<End>	[LINE END]
<Enter>	[RETURN]
<Esc>	[ESCAPE]
<F1>	[HELP]
<F2>	[MENU] SS
<F3>	[MENU] SN
<F4>	[MENU] FB
<F5>	[MENU] FV
<F6>	[MENU] FT
<F7>	[MENU] EN
<F8>	[MENU] EE
<F9>	[MENU] BS
<F10>	[MENU]
<F11>	[T-REG INSERT]
<F12>	Not assigned
<Home>	[LINE BEGIN]
<Ins>	[INSERT TOGGLE]
<Numpad/>	[VISUAL EXIT]
<Numpad*>	[T-REG INSERT]
<Numpad->	[T-REG MOVE]
<Numpad+>	[T-REG COPY]
<Numpad.Enter>	[RETURN]
<PgDn>	[PAGE DOWN]
<PgUp>	[PAGE UP]
<Shft-Del>	[MENU] ET
<Shft-F1>	Not assigned
<Shft-F2>	[MENU] SB
<Shft-F3>	[MENU] SP
<Shft-F4>-<Shift-F8>	Not assigned
<Shft-F9>	[MENU] BR
<Shft-F10>	Not assigned
<Shft-F11>	Not assigned
<Shft-F12>	Not assigned
<Shft-Ins>	[MENU] EP
<Shft-Tab>	[BACKTAB]
<Tab>	[TAB CHARACTER]

F - IBM PC Color Chart

Value	Text on	Background	Value	Text on	Background
16	Black on	Blue	1	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 on	Black	3	Cyan 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 on	Black	5	Magenta 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 on	Black	7	White 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		
14	Yellow on	Black	8	Grey on	Black
30		Blue	24		Blue
46		Green	40		Green
62		Cyan	56		Cyan
78		Red	72		Red
94		Magenta	88		Magenta
110		Brown	104		Brown
126		White	120		White

G - ASCII Table

0	00		16	10	▶	32	20		48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
1	01	☺	17	11	◀	33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
2	02	☻	18	12	↕	34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
3	03	♥	19	13	‼	35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
4	04	♦	20	14	¶	36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
5	05	♣	21	15	§	37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
6	06	♠	22	16	▬	38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
7	07	•	23	17	↨	39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
8	08	◘	24	18	↑	40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
9	09	○	25	19	↓	41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
10	0A	◙	26	1A	→	42	2A	*	58	3A	:	74	4A	J	90	5A	Z	106	6A	j	122	7A	z
11	0B	♂	27	1B	←	43	2B	+	59	3B	;	75	4B	K	91	5B	[	107	6B	k	123	7B	{
12	0C	♀	28	1C	∟	44	2C	,	60	3C	<	76	4C	L	92	5C	\	108	6C	l	124	7C	|
13	0D	♪	29	1D	↔	45	2D	-	61	3D	=	77	4D	M	93	5D	]	109	6D	m	125	7D	}
14	0E	♫	30	1E	▲	46	2E	.	62	3E	>	78	4E	N	94	5E	^	110	6E	n	126	7E	~
15	0F	☼	31	1F	▼	47	2F	/	63	3F	?	79	4F	O	95	5F	_	111	6F	o	127	7F	⌂

128	80	Ç	144	90	É	160	A0	á	176	B0	▒	192	C0	└	208	D0	╨	224	E0	α	240	F0	≡
129	81	ü	145	91	æ	161	A1	í	177	B1	▓	193	C1	┴	209	D1	╤	225	E1	ß	241	F1	±
130	82	é	146	92	Æ	162	A2	ó	178	B2	▓	194	C2	┬	210	D2	╥	226	E2	Γ	242	F2	≥
131	83	â	147	93	ô	163	A3	ú	179	B3	|	195	C3	├	211	D3	╙	227	E3	π	243	F3	≤
132	84	ä	148	94	ö	164	A4	ñ	180	B4	┤	196	C4	─	212	D4	╘	228	E4	Σ	244	F4	⌠
133	85	à	149	95	ò	165	A5	Ñ	181	B5	╡	197	C5	┼	213	D5	╒	229	E5	σ	245	F5	⌡
134	86	å	150	96	û	166	A6	ª	182	B6	╢	198	C6	╞	214	D6	╓	230	E6	µ	246	F6	÷
135	87	ç	151	97	ù	167	A7	º	183	B7	╖	199	C7	╟	215	D7	╫	231	E7	τ	247	F7	≈
136	88	ê	152	98	ÿ	168	A8	¿	184	B8	╕	200	C8	╚	216	D8	╪	232	E8	Φ	248	F8	°
137	89	ë	153	99	Ö	169	A9	⌐	185	B9	╣	201	C9	╔	217	D9	┘	233	E9	Θ	249	F9	∙
138	8A	è	154	9A	Ü	170	AA	¬	186	BA	║	202	CA	╩	218	DA	┌	234	EA	Ω	250	FA	·
139	8B	ï	155	9B	¢	171	AB	½	187	BB	╗	203	CB	╦	219	DB	█	235	EB	δ	251	FB	√
140	8C	î	156	9C	£	172	AC	¼	188	BC	╝	204	CC	╠	220	DC	▄	236	EC	∞	252	FC	ⁿ
141	8D	ì	157	9D	¥	173	AD	¡	189	BD	╜	205	CD	═	221	DD	▌	237	ED	ø	253	FD	²
142	8E	Ä	158	9E	₧	174	AE	«	190	BE	╛	206	CE	╬	222	DE	▐	238	EE	∈	254	FE	■
143	8F	Å	159	9F	ƒ	175	AF	»	191	BF	┐	207	CF	╧	223	DF	▀	239	EF	∩	255	FF	

NOTES: This table displays the IBM PC (OEM) character set.

Windows version: Select **{MISC, ASCII table}** to see how all characters in the currently selected font are displayed.

The values 0 (Null), 32 (Space) and 255 appear identical on the screen. Use **{CONFIG, Characters/Cursors, Null display character}** to change how the Null character is displayed.

INDEX

!

".1$$" file, 345
".1R$" file, 346
"1-END" message, 68
132 Column mode, 352
"<" message, 68
[and] (in regular expressions), 127
\ (regular expression escape character), 128
{ and } (in regular expressions), 130
| (pattern matching codes), 120, 250, 348
| (in regular expressions), 129
|@(r), 125

A

Abandoning file (See also File - Exit and File - Close), 63
<Alt> key shortcuts, 283
ANSI - Translate graphics characters, 110 - 111, 231
ASCII - Translate to EBCDIC, 110 - 111, 231
ASCII Table, 241, 361
Assembly language programming, 76, 124
    Ctags, 178 - 179
Auto-buffering, 346
Auto-configuration (Startup), 57
Auto-execution (of macros), 164
Auto-file save, 66, 285
Auto-indent mode, 135, 270
AUTOEXEC.BAT file, 30

B

[BACKSPACE], 90, 281, 283, 311
[BACKSPACE] emulation mode, 281
[BACKTAB], 311
Backup file (".BAK"), 65, 285, 345
Backward file buffering, 346
Beep level, 294
Beginning of file/buffer, 235
Binary file editing, 214
Binary files, 67, 82, 84, 286
    Header size, 85
BIOS
    Keyboard input, 78, 326
    Screen output, 58, 326
Block, 94 - 109
    Auto-replace, 283
    {BLOCK} menu, 218 - 226
    Characters included, 96
    Columnar blocks, 94, 105
    Copy/move (See Copying/moving text), 94 - 109

Cut & past huge blocks, 103
Delete, 208, 228
Fill, 99, 229
Fill character, 268
Highlighting (See Block markers), 94
Indent, 104, 201
Line blocks, 94
Overstrike, 99
Search, 251, 255
Translating, 110 - 111, 231
Write to disk, 226
Block markers, 67
Auto-cancel, 284
Goto, 236
Highlight color/attribute, 278
Highlighting, 94, 105, 284
Remove, 96, 222, 284, 308
Setting, 94 - 95, 218, 283
Setting columnar markers, 220
Setting line markers, 220
"BLOCK" message, 67
Box drawing mode, 242, 294
Browse mode, 57, 61, 68, 190, 286, 335
Buffer (See Edit buffer), 192
"BYTE" message, 67

C

C programming, 155 - 159
Ctags, 178 - 179
Syntax highlighting, 155
Template editing, 158
[CANCEL], 312
"Caps" lock, 67
<CR>, 74, 287
Carriage-return character, 74, 82, 286
Searching, 122
Case (See Lower and upper case), 76
Center line, 199, 269
CFUNC.VDM file, 175
Clipboard, 70, 103, 197
"COL" message, 67
Color
Chart, 360
Configuration, 277
Edited text, 215, 278
Toggle, 215
Color syntax highlighting - See Syntax highlighting, 155 - 157
Column
Display column #, 67
Go to column #, 237
Columnar block (See Block), 94, 220
Command macros, 161 - 163

Auto-execution, 59, 164
Execute, 163, 244 - 246
Load, 163, 245
Text register usage, 162
Command Mode
Enter via {Escape} menu, 307
Command Mode (VEDIT PLUS), 161, 316
Enter via keystroke macro, 92
Help message, 328
Message on status line, 67
Comments (Assembly language), 124
Compare
Buffers (files), 257
Directories, 170
Files, 168 - 169
COMPARE.VDM file, 168 - 169
COMPDIR.VDM and COMPDIR.BAT files, 170
COMPILE.CNF file, 182
Compiler support, 180 - 184
Description of files, 36
Installation, 181
Startup, 247
Using, 183
CONFIG.SYS file, 30
Configuration
{CONFIG} menu, 263 - 295
Edit buffer dependent parameters, 266
File-type specific, 153 - 154, 271
Initial, 33
Keyboard layout, 38 - 42, 296 - 301
Load from disk, 265
Network, 31 - 32
Overview, 263 - 295
Save to disk, 264 - 265
Startup, 57
STARTUP.VDM file, 149
VEDIT.CFG file, 324
VEDIT.KEY file, 322 - 323
Continuation character, 72
Configuration, 328
Continuation line, 72
Control characters
Display, 74, 213, 275
Entering, 73, 201
Keyboard, 73
Search for, 252
Convert lower to upper case (See Lower and upper case), 76, 227
Copying/moving text, 97
Between files, 141, 168
Huge blocks, 103
To a text register, 99, 206, 224

To the cursor, 222
To the Scratchpad, 98, 206
To the Windows clipboard, 103, 197
Ctags facility, 178 - 179
<Ctrl> key shortcuts, 283
<Ctrl-\>, 67
<Ctrl-C>, 165, 340
<Ctrl-N>, 252
<Ctrl-Z>, 325
Current directory, 142, 305
Cursor
Blink rate, 276
Display (style and type), 78, 276
Position after block insertion, 202, 284
Positioning mode, 280
[CURSOR DOWN], 312
[CURSOR LEFT], 281 - 282, 312
Cursor movement
By paragraphs, 48, 314
By screens/pages, 48, 314
By sentence, 352
By words, 47, 314
In text, 47, 312, 327
[CURSOR RIGHT], 281 - 282, 312
[CURSOR UP], 312
Cut and Paste (See also Copying/moving text), 206

D

Data files (See also Binary files), 84
Database files
Editing, 85, 173 - 174
Date
Display, 305
Insert date and time, 89, 352
dBase files, 173 - 174
Default directory, 142, 305
[DEL NEXT WORD], 47, 209
[DEL PREV WORD], 47, 209
Delete
Block of text, 208, 228
Large blocks, 81
Line, 209
Text (Overview), 46
Undo, 81, 204
Words, 209, 312
[DEL NEXT WORD], 313
[DEL PREV WORD], 312
[DELETE], 283, 312
At end of line, 83
Detabbing (Tabs to spaces), 228
Directory
Compare two directories, 170

Current (Starting), 142
Default, 305
Disk full error (recovery), 338
“DISK” message, 67
Disk space
Maximum file size, 346
Usage, 346
Display font, 217
Display modes, 74, 213, 275
DOS Shell, 242, 244
Enable/disable, 328
DOS text file, 83, 286
Convert to UNIX, 83
Drag and drop, 55, 138
Duplicate line (macro), 352

E

EBCDIC, 67
Translate to ASCII, 110 - 111, 231
“EBCDIC” message, 67
Edit buffer
Attach to window, 144
Close, 140, 189
Configuration parameters, 266
Details, 143
ID number, 274
Insert as a text register, 143
Multiple file editing, 60
Naming, 144
Number on status line, 67, 192
Open, 186 - 188
Switching, 140, 192 - 193
Usage, 303
Edit function
Reference, 311 - 316
{EDIT} menu, 196 - 202
Edit session restore, 57, 64, 194, 285
Editing
Binary files, 214
Hex-mode, 79 - 80, 214
Multiple files, 60, 138 - 143, 187 - 188
New file, 186 - 188
One file in two windows, 146
Switching between files, 140, 192 - 193
Emulation
Modes, 77
Of other editors, 280
End-of-file character (<Ctrl-Z>), 325
End-of-file processing, 325
End-of-file/buffer, 235
End-of-line character (See Newline character), 82
[ENTER CTRL], 201, 313

<Enter> key, 45, 83, 281, 286
Entering new text, 45
Environment variable “VEDPATH”, 329
Environment variable “VBACKUP”, 66
Environment variable “VEDIT”, 60
Erase (See Delete), 209
[ERASE BOL], 209, 313
[ERASE EOL], 209, 313
[ERASE LINE], 209, 313
Error messages, 331 - 342
<Esc> key
    Assignment to [ESCAPE], 307
    Enter into text, 201, 297
    Usage with UNIX, 39
[ESCAPE], 313
{ESCAPE} menu, 307 - 309
Event macros, 158 - 159
Exit
    Save/restore edit session, 64, 194
    Screen restore, 328
    VEDIT, 54, 63 - 66, 194, 309

F

File
    Altered/unaltered, 63, 68, 190, 203
    Binary, 82 - 85, 286
    Buffering, 346
    Change name (Save as), 191
    Close, 140, 189
    Comparison, 257
    Comparison (macro), 168 - 169
    Exit - save/abandon, 63, 194, 309
    {FILE} menu, 186 - 195
    Goto position (offset), 237
    Handling/management, 345 - 347
    Input file, 305, 345
    Inserting, 202
    Large (long), 346
    Locking, 347
    Maximum file size, 346
    Names of files being edited, 67, 304
    Open, 187 - 188
    Open/close event macro, 271
    Output file, 305, 345
    Position, 237
    Read-only, 190, 335
    Save and continue editing, 54, 190
    Save and exit, 54
    Size, 305
    Switching between files, 140, 192 - 193
    Type (Binary/text), 82 - 85, 286
File-type specific configuration, 153 - 154, 271

Filename (displaying), 67
Fill (See Block - Fill), 229
Font
    Display, 69, 217
    Printer, 113
Form-Feed character, 290
Format paragraph, 136, 199
Free memory space, 305
Full disk (See Disk full error), 338
Function/control keys, 295, 326

G
{GOTO} menu, 235 - 240
Graphics characters
    Display, 74, 213, 275
    Entering, 73
    Keyboard, 78
    Search for, 252
    Strip high bit, 230

H
Help, 313
    Help level, 294
    {HELP} menu, 302 - 306
    Introduction, 302
    On-line, 302, 313
    Status display, 304
Hex-mode editing, 79 - 80, 214
Hexadecimal
    Offset into file, 68
High bit characters (See Graphics characters), 73
Highlighting (See Block markers)
    Color/attribute, 278
Horizontal scroll increment, 274
Horizontal scroll margin, 72, 85, 274
Horizontal scrolling, 71, 274
HTML editing, 155 - 160

I
Icon properties - changing, 56
Indent increment, 135, 271
Indenting text, 104, 134, 200
Input file, 305, 345
“INS” message, 68
Insert
    File, 202
    From ASCII Table, 241
    Text (Overview), 45
Insert mode, 210, 276, 281, 325
[INSERT TOGGLE], 210, 313
Installation, 21
    DOS, 24 - 29
    DOS version in OS/2, 28
    DOS version in Windows, 27

Files (Description) 34 - 37
IBM PCDOS/MS-DOS, 24
Network, 31 - 32
Testing, 29
Windows, 22 - 23
Invoking VEDIT, 147 - 150
Options, 57, 164
Overview, 44, 55 - 62

J

Java SDK support, 181
Justifying paragraphs, 136 - 137, 270

K

Key conversion character, 76, 271
.KEY file (VEDIT.KEY), 300, 322 - 323
KEY-MAC.LIB file, 89, 92
Keyboard
Notation, 19
Options, 78, 295, 326
Polling, 26, 58
Shortcuts, 283
Typematic rate, 26
Keyboard layout, 38 - 42, 296 - 301
Change, 91 - 93, 299, 322 - 323
Display/Help, 303
Edit, 91 - 93, 299
Edit (Keyboard layout file), 323
Load from disk, 93, 301
“Normal” layout, 39 - 40, 356 - 359
Print, 299
Save to disk, 300
Supported keys, 326
UNIX, 39
Unused keys, 299
Keystroke macros, 86 - 90
Add/Record, 88, 297 - 298
Built-in, 41
Delete (un-assign a key), 90
Escape sequence, 296
Examples, 88 - 89, 351
Help, 303
Hot-keys, 86
Macro language (VEDIT PLUS), 86, 89, 92, 161
Misc. notes, 297, 351
Modify, 90
Repeating, 297
Save, 300

L

Large files (See File - Large), 346
Left margin, 134, 269, 288
Line
Center, 199, 269

Display line #, 67, 85, 286
Editing long lines, 72, 85
Erase, 209
Go to line #, 236
Split into two, 45
Toggle 25/28/50 line mode, 215
Wrap long lines on screen, 72, 274
[LINE BEGIN], 47, 282, 313
Line block (See Block), 94, 220
Line emulation mode, 282
[LINE END], 47, 282, 314
“LINE” message, 68
<LF>, 74, 286
Line-Feed character (See also Newline character), 74, 82, 286
Lower and upper case
Change/switch case of text, 227
Converting keys, 76, 271
Reverse case (Keyboard), 78, 295
Searching, 250, 257, 292

M

Macintosh text file, 83, 286
Convert to DOS, 84
Macro language (See Command Macros), 161
Macros (See Command macros or Keystroke macros), 161 - 163
Margin (See Left and Right margins), 134
Markers (See Block markers or Text Markers), 238
Match parentheses, 239
Maximum file size, 346
Memory
Restricting usage, 59
Space free, 305
[MENU], 314
Menu system
Preview, 263
Microsoft Windows, 27
{MISC} menu, 241 - 248
Monochrome screen colors, 29, 58
Mouse
Right-click menu, 310
Mouse support
Zoom button, 211
Multi-Tasking Operating Systems, 347
Multiple drives, 346
Multiple file processing, 166 - 167
Multiple files (See Editing), 138 - 143

N

Network
Configuration, 31
File locking, 347
Installation, 31 - 32
Printing, 291

"New file" message, 57
Newline character, 45, 82, 286
    Display, 74, 276
    Searching, 121 - 122, 252
[NEXT LINE], 314
[NEXT PARAGRAPH], 48, 314
[NEXT TAB STOP], 314
[NEXT WORD], 47, 314
Notation, 19
Null character, 73, 122, 129, 252
    Display, 276
"Num" lock, 67
Numeric expressions, 70

O

Offset into file, 68, 237
Offset paragraph, 136
OS/2, 28
Outliner (CFUNC macro), 175
Output file, 305, 345
Overstrike mode, 45, 276, 281, 325
Overwrite-only mode, 62, 100, 286

P

Padding (Tabs and spaces), 105, 135
[PAGE DOWN], 48
[PAGE UP], 48, 314
    Number of lines/overlap, 327
Page eject, 291
Paragraph
    Definition of, 133
    Formatting, 136, 199, 269
    Formatting - Enable, 269
    Justifying, 136 - 137, 270
    Offset, 136
    Unjustify, 137
Parentheses matching, 239
Pattern matching, 120, 348
    Codes, 120
    Pattern sets, 124
"POS" message, 68
[PREV PARAGRAPH], 48, 314
[PREV WORD], 47, 314
Print
    Basic operation, 114
    Block, 114, 117, 193
    Configuration, 288
    Control characters, 116, 289
    Dialog box, 193
    Display mode, 116
    EBCDIC, 116
    Eject Page, 291
    Entire file, 114 - 115, 165, 193

Finish print job, 291
Fonts, 119
Form-Feed character, 290
Formatter (macro), 115, 165
Hexadecimal, 116, 289
Introduction, 53, 113 - 119
Job, 290
Laser Printer notes, 288
Line spacing (double, etc.), 289
Macro, 115, 165
Margins, 115, 288
Mode, 289
Paper length, 288
Print-job, 117
Printer port selection, 291
Start/Finish strings, 117 - 118
To file, 291
Troubleshooting, 115
Wrapping long lines, 289
PRINT.VDM file, 115, 164 - 165
Programs
Run from within VEDIT, 176 - 177

Q

Quit (abandon) and exit (See File - Exit), 63

R

Read-only file, 190, 335
Read-only mode, 57, 286
Record mode, 84
Record size, 84
Redo, 49, 81, 204
Registers (See Text register), 98
Regular expressions, 126, 292
[REPEAT], 196, 295, 314
Keystroke macros, 297
[REPEAT LAST], 314
Repeating operations (See also [REPEAT]), 50, 196
Replacement string, 253
Restore edit session, 57, 64, 194, 285
Retabbing (Spaces to tabs), 228, 267
[RETURN] (See also <Enter> key), 315
Right margin, 135, 269, 289
Right-click menu, 310
“RM” message, 68
Running programs (Shell), 176-177, 242, 244, 328

S

Save configuration changes into VEDIT, 265
Save configuration changes to disk, 264
Save keyboard layout, 300
Save text and continue, 54, 190
Save text and exit, 54, 63, 194, 309
Scratchpad (text register), 70, 98, 206

Screen
Color chart, 360
Color/attributes, 58, 277 - 278
Display (updating), 78
Display modes, 74, 213, 275
Initialize, 212
Overlap when paging, 327
Restore on exit, 328
Scrolling, 71 - 72
Size, 327, 352
Writing (Direct Memory/BIOS), 58, 326
[SCREEN BEGIN], 282, 315
[SCREEN END], 282, 315
Scroll bars, 216, 273
[SCROLL DOWN], 71, 282, 315
[SCROLL LEFT], 71, 274, 315
[SCROLL RIGHT], 71, 274, 315
[SCROLL UP], 71, 282, 315
Scroll increment, 274
“Scroll” lock, 67
Scroll margin, 274
Scrolling, 71 - 72, 315, 327
Search, 120 - 132, 249 - 257
Again, 254
Blocks, 251, 255
Carriage-Return, 122
Case sensitive, 250, 257, 292
Configuration, 292
Control character, 252
Error, 293
From beginning of file, 251
Incremental search, 254
Introduction, 249
Local (only to end of memory), 251
Modes, 120, 250, 292
Multiple files, 166, 256
Newline, 122
Next, 254
Null character, 122, 129
Options, 250
Pattern matching codes, 348 - 350
Previous, 254
Regular expressions, 250, 292, 349
{SEARCH} menu, 249 - 257
Simple mode, 250, 292
String, 125, 249
Text, 249
Words, 251
Search and Replace, 120 - 132, 249 - 257, 282
Again, 254
Multiple files, 167

Options, 253
Serial number, 306
Set markers (See Block markers or Text markers), 218
Setup.exe, 34

Shelling out to DOS, 176-177, 242, 244, 328
<Shift> key shortcuts, 283
Shortcuts, 69
SORT.VDM macro, 171 - 172
Sorting, 112, 171 - 172, 231
Spaces
Convert to tabs, 105, 109, 228, 267
In paragraphs, 137, 270
Trailing, 105
Trimming, 108, 268
Starting VEDIT, 44, 55 - 62
Startup configuration, 57
STARTUP.VDM file, 58, 147 - 150, 326
Status display, 304
Status line, 67 - 68
Configuration, 327
Strip comments (assembly language), 124
Strip high bit (Bit 8), 230
Substitute (See Search and Replace), 120 - 132
Switching buffers (See Edit buffer - Switching, 140
Syntax highlighting, 155-157, 272
Colors, 279
Load syntax file, 247

T

[T-REG COPY], 224, 316
[T-REG INSERT], 225, 316
[T-REG MOVE], 224, 316
[TAB CHARACTER], 267, 281, 315
Tab character (key), 75, 267, 315
Convert spaces to tabs, 105, 109, 228
Convert Tab character to spaces, 105, 109, 228
Display, 74, 76, 276
Expand Tab key with spaces, 76, 267
Tab stops, 75, 267
Technical support, 20, 306
Replacement disk, 21
Template editing, 158 - 159, 272
Load template file, 246
Temporary disk file (".1R$"), 346
Text Markers
Go to, 239
Setting, 238
Text register
Copy/move to, 99, 206, 224
Display, 304
Emptying, 100
In search string, 125, 252

Inserting, 99, 207, 225
Memory usage, 304 - 305
Naming, 101
Overview, 98
Usage, 100, 304
Usage by command macros, 162
Time
Display, 305
Insert date and time, 89, 352
{TOOLS} menu
Description, 151
Load menu, 248
Toolbar, 216, 273
Trailing spaces, 105
Translating a block/file, 110 - 111, 231
Transpose letters, 352
Troubleshooting, 354 - 355
Configuration and Startup, 320 - 321
{Tutorial} menu, 248

U

Undenting - See Indenting, 200
Undo
Deletion, 81, 204, 229
Edit, 203
Introduction, 49, 81, 203
Levels, 81, 328
Line, 203
Redo, 81, 204
Reset, 205
{UNDO} menu, 203 - 205
UNIX text file, 83, 286
Convert to DOS, 83
Unjustify (See also Justifying paragraphs), 137
Upper and lower case (See Lower and upper case), 76
User Config Directory, 318, 320
User interface, 69 - 70
{USER} menu
Description, 151
Load menu, 248

V

V-SPELL
Introduction, 16
Run from inside VEDIT, 351
V-SWAP
DOS Shell, 243
Enable, 286
Installation, 30
".VDM" file, 162
"VEDIT" environment variable, 60
VEDIT Home Directory, 318, 320
VEDIT icon

Properties, 56
VEDIT.CFG file, 149, 317 - 330
VEDIT.INI file, 21, 58, 66, 150, 318, 320
VEDIT.KEY file, 300, 319 - 320, 322 - 323
“VEDPATH” environment variable, 329
Version number (VEDIT), 304, 306
Vertical scrolling, 71
VGA display, 215, 352
View - See Windows, 258
{VIEW} menu, 211 - 217
Virtual space mode, 276
[VISUAL ESCAPE], 316
[VISUAL EXIT], 92, 316
Visual Mode, 316
Definition, 161

W

Wildcard characters, 122, 126, 166
WILDFILE.VDM file, 166 - 167
Windows, 144 - 146
Arrange icons, 259
Attach to buffer, 144
Border characters, 325
Borders, 273
Cascade, 258, 274
Close, 140, 260
Color/attributes, 278
Command Mode window, 308
Delete, 260
Display modes, 74, 213, 275
Editing multiple files, 139, 189
Full-size, 139-140, 146, 212, 274
ID Number/name, 144, 260, 274
Minimized, 259
Multiple per file, 259
Name display, 274
Naming, 144
Overview, 144 - 146
Remove, 260
Reset, 139, 212
Resize, 139
Splitting, 139, 259
Switching, 145, 261 - 262
Tile, 258
{VIEW} menu, 211 - 217
{WINDOW} menu, 258 - 262
Zooming, 139, 145, 211
Windows Clipboard, 103, 197
Word
Count, 351
Definition of, 133, 330
Selecting as a block (mouse), 221

Wrap (See also Right margin), 135
Word Perfect keyboard layout, 93
Word Processing, 133 - 137
Configuration, 269
Word wrap - Enable, 269
WordStar keyboard layout, 78, 93

Z Zooming windows (See also Windows), 145