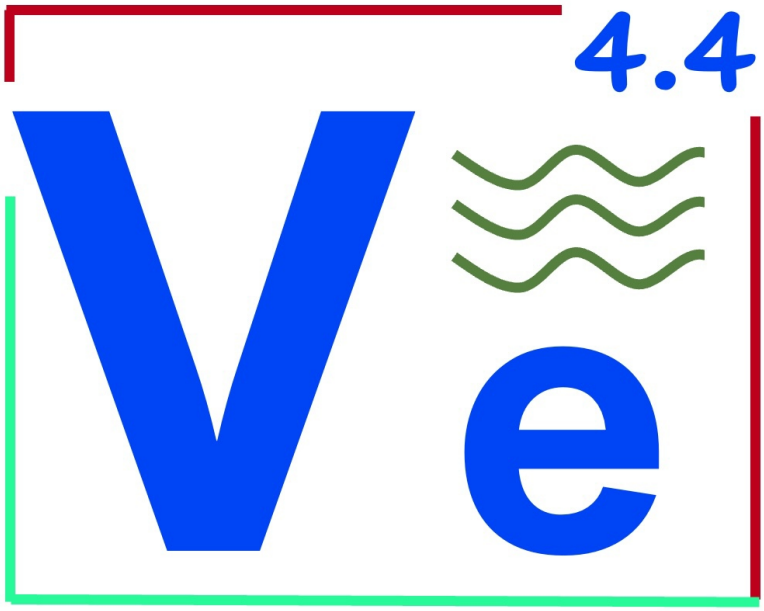


DOS VE v4.4a User Manual

June 2023

VE Brought to You By:
CampbellWare (www.inverary.net)
1984, 1994, 2005, 2023

This document uses Courier, Times and Bitstream Vera Sans fonts.
If these fonts are not installed, this document may not display properly.

(This document prepared using LibreOffice 4.0.6)

Table of Contents

1.0 Why VE? ...an Editorial....6
2.0 Introduction....7
2.1 A Brief History of VE....7
2.2 VE Key Features....7
2.3 VE Has Been Verified On....8
2.4 VE Website....8
2.5 VE System Requirements....9
2.6 Installing VE....10
2.7 Acknowledgments....10
3.0 VE Quick Start Guide....11
4.0 Running VE....12
5.0 Multiple Screen Resolution Support....13
5.1 Running VE at 80x50 (VGA Only)....13
5.2 Running VE at 80x43 (EGA and VGA)....13
5.3 Running VE at 80x25 (MDA, CGA, EGA, VGA)....13
5.4 Running VE From Non 80x25 Resolutions....13
6.0 Graphics Types and VE....14
6.1 MDA, Hercules, CGA, EGA and VGA....14
6.2 VE Graphics Adapter Support....14
6.3 VE "-g" Startup Option....14
7.0 VE User Interface....15
7.1 Entering Text....15
7.2 Entering VE Commands....15
7.3 Function Keys (F1-F12)....16
7.4 Exiting From Menus, Prompts, Etc....17
7.5 Command Repetition Counts....17
7.6 Repeating Commands Sequentially....17
7.7 Mouse Support....17
7.8 Getting Help....18
8.0 VE Screen Layout and Interaction Model....20
8.1 VE Screen Shot....20
8.2 VE Screen Layout Notes....20
8.3 Status Line Command Interaction Model....21
8.4 Status Line Response Line Editing....21
8.5 VE Open File Dialog....22
8.5.1 Selecting Files Interactively....22
8.5.2 Selecting Files Using the Mouse....23
8.5.3 Selecting Files by Typing Their Name....23
8.5.4 Selecting Files Using Filename Auto-Completion....23
9.0 A Few Words About....25
9.1 A Few Words About Tab Handling....25
9.2 A Few Words About Text File Formats....25
10.0 Entering Text....26
10.1 Entering Text....26
10.2 Breaking and Joining Lines....26
11.0 Formatting Text....27
11.1 On-The-Fly Text Wrapping....27
11.2 After-The-Fact Text Wrapping....27

12.0 Deleting, Copying and Pasting Text..... 29
12.1 Deleting Text..... 29
12.1.1 Deleting Characters with the Delete Key..... 29
12.1.2 Deleting Characters with the Backspace/Rubout Key..... 29
12.1.3 Deleting Multiple Characters with Delete, Backspace..... 29
12.1.4 Deleting All Characters to the Right, Left..... 29
12.1.5 Deleting Words..... 30
12.1.6 Deleting Lines..... 30
12.1.7 Deleting Paragraphs..... 30
12.1.8 Deleting Blocks of Lines..... 30
12.1.9 Deleting Portions of Lines..... 30
12.1.10 Deleting Rectangular Blocks of Text..... 31
12.2 Copying Text..... 32
12.2.1 Copying Lines..... 32
12.2.2 Copying Portions of Lines..... 32
12.2.3 Copying Blocks of Lines..... 32
12.2.4 Copying Paragraphs..... 33
12.2.5 Copying Rectangular Areas..... 33
12.3 Pasting Text Within and Between Files..... 33
12.3.1 Pasting Lines..... 34
12.3.2 Pasting Portions of Lines..... 34
12.3.3 Pasting Blocks of Lines..... 34
12.3.4 Pasting Paragraphs..... 35
12.3.5 Pasting Rectangular Areas..... 35
12.3.6 Intelligent Paste (VE's Global Paste Command)..... 35

13.0 Moving Around In A File..... 36
13.1 Scroll Screen Up One Page, Down One Page..... 36
13.2 Scroll Screen Half Page Up, Half Page Down..... 36
13.3 Scroll Screen Down by an Arbitrary Amount..... 36
13.4 Move Selected Line to Top of Screen..... 36
13.5 Scroll Screen Half Right or Half Left..... 36
13.6 Scroll the Screen Right by an Arbitrary Amount..... 36
13.7 Move Selected Column to Left Edge of Screen..... 37
13.8 Move Cursor Left, Right, Up and Down..... 37
13.9 Move Cursor to Start of Line, End of Line..... 37
13.10 Move Cursor to Left, Middle, Right of Display..... 38
13.11 Move Cursor to Top, Center, Bottom of Display..... 38
13.12 Move Cursor Using Extended Cursor Motion..... 39
13.13 Move Cursor To Display Edge..... 40
13.14 Move Cursor to Next Word, Previous Word..... 40
13.15 Move Cursor to Next Para, Previous Para..... 40
13.16 Move to Top of File..... 40
13.17 Move to Bottom of File..... 40
13.18 Jump Start, End, Line Number..... 41
13.19 Tags, and Jumping to Tags..... 42
13.20 Current Line, Column Information..... 42

14.0 Finding and Replacing Text..... 43
14.1 Finding Text..... 43
14.2 Replacing Text..... 43
14.3 Repeating Find/Replace Operations..... 44

15.0 Operating on Blocks of Lines..... 45
15.1 Line Oriented Copy, Cut and Paste..... 45
15.2 Defining a Line Block..... 46
15.3 Line Block Copy, Cut, Paste, Wrap, Read, Write..... 46
15.4 Repeating Line Block Commands..... 48
15.5 VE Block Paste Command Revisited..... 48

16.0 Operating On Column Areas (Rectangles)..... 49
16.1 Defining a Column Area (Rectangle)..... 49
16.2 Column Area Insert, Copy, Cut, Paste, Move, Read, Write..... 50
16.3 Repeating Column Area Commands..... 52
17.0 Inserting From, Writing to External Files..... 53
17.1 Writing Selected Areas to an External File..... 53
17.2 Inserting an External File into File Being Edited..... 53
18.0 Exchanging Lines and Rectangles..... 54
19.0 Undoing Changes..... 55
20.0 Saving Your Work..... 56
20.1 Saving Your File As You Work..... 56
20.2 Saving Your Work To Another File..... 56
20.3 Saving Your Work and Exiting..... 56
20.4 Saving Your File via the Goto File Dialog..... 56
20.5 Knowing When to Save - Change Tracking..... 57
21.0 Editing Multiple Files..... 58
21.1 Multiple File Support Overview..... 58
21.2 Opening Multiple Files From the Command Line..... 58
21.3 Opening Additional Files from Within VE..... 59
21.4 Discarding Files from Within VE..... 59
21.5 Moving Between Open Files Sequentially..... 59
21.6 Inserting an External File into File Being Edited..... 60
21.7 Moving Between Open Files via Goto File Dialog..... 60
21.8 Saving and Discarding Files via Goto File Dialog..... 62
22.0 Exiting from VE..... 63
22.1 Quitting VE from the Quit Command..... 63
22.2 Quitting VE from the Goto File Dialog..... 63
23.0 VE Macros..... 64
23.1 VE Macros Overview..... 64
23.2 On-the-Fly Macros..... 64
23.3 Creating a Regular Macro..... 64
23.4 Executing a Macro..... 65
23.5 Repeated Macro Execution..... 65
23.6 Undoing a Macro..... 65
23.7 Listing the Available Macros..... 65
23.8 Deleting Macros..... 65
23.9 Saving Macros to Disk..... 66
23.10 Loading Macros from Disk..... 66
24.0 Miscellaneous Commands..... 67
24.1 Redrawing The Screen..... 67
24.2 Changing Case..... 67
24.3 About VE..... 67
24.4 Shell to DOS..... 68
24.5 A Look At VE's Internal Status..... 68
25.0 Configuring VE..... 69
25.1 Configuring VE's Color Schema..... 69
25.2 Configuring Auto Indent..... 70
25.3 Configuring Tabs..... 70
25.4 Configuring Find/Replace's Start/End Match Characters..... 70
25.5 Configuring Text Wrapping..... 70
25.6 Configuring Text File Format..... 71
25.7 Configuring VE CPU Performance Consumption..... 71
25.8 Saving The VE Options Configuration..... 72
25.9 Loading The VE Options Configuration..... 72

26.0 Miscellany....73
26.1 Repetition Counts Default Commands to Unmodified Form....73
26.2 The “*” Repetition Count Isn't Actually Infinity....73
26.3 VE File Writes Overwrite “filename~”....74
27.0 Known Issues....75
27.1 Open/Goto File Dialogs and Macros....75
27.2 Auto Indent and On-the-fly Text Wrapping....75
27.3 On-the-fly Text Wrapping and Justify/Delete....75
27.4 Support for Text Files with Lines >384 Chars....75
27.5 Screen/File Mismatch....76
28.0 Bug Reports, Donations, Gratuitous Praise....77
29.0 Appendix A – VE Keyboard Commands....78
29.1 About VE Keyboard Commands....78
29.2 PC Extended Keyboard Key Commands....78
29.3 Function Key Commands....78
29.4 ESC Key Commands....79
29.5 ALT Key Commands....82
29.6 CTL Key Commands....84
30.0 Appendix B – A Brief VE Release History....85
30.1 Release History....85
30.2 New in Release 4.x....86

1.0 Why VE? ...an Editorial

There are a lot of text editors out there. Why VE?

VE is a full screen DOS/Windows text editor designed for high speed, small executable size and a rich feature set. VE accomplishes all these things and packs a powerful punch into its slimline 128K executable. Why VE? Think **vi** power, but smaller, faster and friendlier to use.

In the full screen text editor space, **vi** is perceived to be the golden standard, the product to beat. VE goes head-to-head with **vi** and improves on it in a really fundamental way. How many times have you fired up **vi** and started to type in your favorite file, only to find that you forgot to put vi into text entry mode and you have just executed a random sequence of commands that has completely trashed your file? Thank goodness you can just exit out and start again! VE stands this paradigm on its head. Instead of always being in command mode unless you execute a command to enter text entry mode, VE is always in text entry mode unless you execute a command, and then it automatically returns to text entry mode after completing the command! VE is almost always in text entry mode. In this most basic and simple of ways, VE dramatically derisks the task of entering and editing text in a file.

VE is friendlier to use than **vi** as well. Keyboard-based text editors can be confusing to use until you learn their command key sequences. Think of emacs as the ultimate example of this sort of steep learning curve. VE recognizes this and provides two levels of built-in task-oriented help and of course a comprehensive user manual, which is included in the software distribution. The first level of built-in help is three screens of VE command key summaries, always available via ALT-h (Help). The second level is an extensive set of tips, one topic per tip. These are "on demand" vs. popping up when you start VE and are always available via ALT-t (Tips). Conveniently, if you want to read multiple tips in one sitting, just press the "n" (next) or "p" (previous) keys while viewing a tip. Best of all, you never have to remember the ALT-h and ALT-t command sequences. VE makes them part of its main screen layout, almost always shown on the top line of the display.

VE feels natural to use. Ease of use has always been an important VE design consideration, so while you are entering text, the PC extended keyboard cursor keys do exactly what you would expect them to do, as do the Home and End keys respectively (move to the start of a line, move to the end of a line). Ditto for the Delete key. Usage pseudo-standards like CTL-c, CTL-x, CTL-v and CTL-s all do what you would expect as well (copy, cut, paste and save, respectively). However, this is where a little bit of VE magic kicks in. Many of these keys can be modified with the ALT key, allowing them to perform modified versions of the expected functions. For example, ALT PgUp move half a page up, ALT PgDn moves half a page down, ALT Cursor Right scrolls the display half a screen to the right and ALT Cursor Left reverses that, scrolling the display half a screen to the left. Not to be left out, ALT-Delete deletes words vs. characters.

Many command keys can also be combined with the ESC key to produce different, but related, results from the underlying key function. ESC-PgUp moves to the top of the file; ESC-PgDn moves to the end of the file. ESC-Delete also deletes words vs. characters (identical to ALT-Delete). ESC-Cursor Right moves the cursor to the right by one word while ESC-Cursor Left moves the cursor to the left by one word. ESC-Cursor Up moves the cursor up by one paragraph, and of course ESC-Cursor Down moves the cursor down by one paragraph.

VE supports mouse functionality! VE was designed from the ground up to be keyboard-based, not menu/mouse driven. VE is a full screen text mode editor, not a GUI editor. However, VE provides intelligent mouse support as an increment to its keyboard interface. Consequently, you can position the cursor anywhere on screen by just moving the mouse pointer to the desired location and clicking. When you don't want the mouse pointer on screen, it can be hidden with the ALT-m keystroke. That same keystroke will retrieve it when you next want it. Need to select a word? A simple double click anywhere in the word will accomplish the desired result, highlighting that word and starting the Column/Rectangle command to operate on the result. The same goes for entire rectangular regions, where you can click and drag to select, or left click on the upper left edge and right click on the lower right edge to select the region in the Column/Rectangle command. Finally of course, you can use the mouse in a "point and shoot" manner to select files in any full screen file selection dialog, such as the Open File dialog.

Finally, VE comes in three versions: two 16-bit versions and a 32-bit version. The 32-bit version has a larger executable size, but is able to edit much larger files, more files, or both. The 16-bit versions are smaller, faster and suitable for most routine editing tasks. No matter which you choose, the feature set and the user experience are the same across all three.

2.0 Introduction

2.1 A Brief History of VE

VE (Visual Editor) is a small, fast, feature-rich text editor with strong multi file editing support. VE started life in 1984 in Toronto Canada as a text editor for the NABU 1600 small business computer - an Intel 8086-based machine with 512Kb of RAM and a 10Mb hard drive, delivered with the QUNIX operating system.

The name "VE" was a good natured jab at the vi editor, which was delivered with the NABU 1600, but which would never run properly on it (termcap issues), necessitating the creation of VE. For several years in the mid 1980's, VE was the sole text editor for a small community of users on the NABU 1600.

In 1994, the author acquired a PC and ported VE to MS-DOS and VGA. Through the mid 1990's, VE was in use by a small community of MS-DOS users.

In 2005, the author ported VE to Linux (various distributions) and ncurses, added significant new functionality, and released it to the Linux user community as VE v3.5. This enhanced Linux version of VE was then ported back to DOS and released as DOS VE 3.5g.

In 2022/2023, the author again enhanced VE, adding new features, enhancing existing ones, optimizing performance and resolving several outstanding bugs. The result was released to the PC enthusiast community as DOS VE 4.0c in 2022 and DOS VE 4.4a in 2023.

VE is provided free of charge for all to use and enjoy.

2.2 VE Key Features

Key features of DOS VE 4.4a are:

- Small, fast, tight implementation
- Strong large file and multi file support
- Horizontal scrolling for files wider than display/window width
- Support for chars, words, paragraphs, lines, line/rectangular text areas, files
- Support for both line-oriented and column oriented (rectangular) text areas
- Interactively defined and “on-the-fly” macros
- Universal undo
- Built-in real time help and on-demand tips
- Support for MDA/Hercules (80x25), CGA (80x25), EGA (80x25, 80x43) and VGA (80x25, 80x43 and 80x50)
- Support for PC, PC XT, PC AT and onwards

2.3 VE Has Been Verified On...

Software: the DOS versions of VE (ve88.exe, ve.exe and ve32.exe) have been verified on MS-DOS 3.30, IBM PC-DOS 3.30, MS-DOS 5.0, MS-DOS 6.22, on the Windows for Workgroups 3.11 DOS prompt window and on the Windows 95 DOS prompt window. Verification was performed with the standard MS-DOS command.com command interpreter, JP Software's 4DOS command interpreter and finally Norton Utilities 8.0 NDOS command interpreter.

Hardware: VE has been verified on a 4.77 MHz PC XT 8088 clone, two EGA-equipped PC AT clones, 10 MHz and 12 MHz 286 respectively, VGA-equipped 486 PCs ranging from 33 MHz (486DX-33) up to 100 MHz (486DX4-100) and on 2nd generation Pentium PCs (90 MHz only). Sanity tests have also been performed on 200MHz Pentium Pro and 450 MHz Pentium II PCs.

Emulators: VE has been verified on DOSBox-X 0.83.23 running on Windows 10 and on 86Box 3.11 running on macOS 12.6 (Monterey).

2.4 VE Website

Please visit www.inverary.net/ve for the latest information on and releases of VE.

2.5 VE System Requirements

Software:

DOS VE requires MS-DOS 3.30 or IBM PC-DOS 3.30, or higher releases of each.

Graphics Hardware:

VE requires MDA, Hercules, CGA, EGA or VGA graphics.

Almost all PCs, from the first PC and PC XT models onwards support one of MDA, Hercules, CGA, EGA or VGA and so can run VE.

CPU Hardware:

VE requires an 8088 or later CPU.

All PCs, from the first PC and PC XT models onwards, use an 8088 or later x86-family processor and so can run VE.

VE is distributed as both 16-bit and 32-bit executables in order to provide optimized CPU-specific support. The two 16-bit executables (ve.exe, ve88.exe) are targeted to different x86 processors. **ve.exe** is compiled for 80186/80286 (PC AT and later) while **ve88.exe** is specifically compiled for 8086/8088. The 32-bit executable (**ve32.exe**) is compiled for 386 and later CPUs, VGA graphics and extended memory.

Practically speaking, **ve.exe** will be the most commonly used VE executable, since it supports the 80286 CPU and all x86 CPUs that came after it. **ve88.exe** is provided specifically to deliver 8088-based PC and PC XT support. However, since the 8088 is the lowest common denominator in the x86 family, **ve88.exe** will run on all x86 CPUs from the 8088 right through the Pentium line. **ve32.exe** provides 32-bit support (via DPMI) for PCs that are equipped with a 386 or later CPU and extended memory.

Keyboard:

VE does not require any specific keyboard.

However, VE has only been tested using the now standard 101-key PC Extended Keyboard introduced with the IBM PC AT. The keycodes returned by this keyboard are hard coded into VE and cannot be customized by users.

VE will operate successfully with other keyboards, but many key bindings may not work as intended – on earlier keyboards, it is likely that the keycodes for one or more keys may not be the same as they are on a PC Extended Keyboard. Accordingly, a standard PC Extended Keyboard is recommended for users of VE.

2.6 Installing VE

Installing VE:

VE is a self contained executable. Installation is extremely simple. Unzip the release zip file (which you have done successfully if you are reading this document) and cd into the ve-4.4a directory that is created. Simply copy the executable files ve.exe, ve88.exe and ve32.exe to any convenient directory on your PATH, and you are ready to use VE.

For an enhanced experience, also copy the file ve.tps to C:\. This file is VE's tips database and enables the ALT-T (Tips) command. You will find many useful and work-saving VE tips and tricks by reading the supplied tips.

ve88.exe, ve.exe or ve32.exe?

If you have an 8088 or 8086 processor (typically an IBM PC or PC XT) you will need to use **ve88.exe**, which is compiled specifically for the 8086/8088 opcode set. ve.exe is compiled for 80186/80286 instructions and thus will not execute on the IBM PC and PC XT.

If you have an 80286 processor (PC AT) or later, you will want to use **ve.exe**, which is compiled using 186 and 286 real mode opcodes. As a result of these additional opcodes, ve.exe is the smallest and fastest of the VE executables and will be the most commonly used VE variant.

If you have a 386 or higher processor, you may wish to use **ve32.exe** instead of ve.exe. ve32.exe takes advantage of the larger addressing space of the 386 and higher processors, allowing the editing of larger files, more files simultaneously, or both, than ve.exe. The trade off is that as a result of its 32-bit targeting, ve32.exe is larger and slower to load than ve88.exe and ve.exe. Although you might think so, ve32.exe's 32-bit targeting does not give it access to Extended memory. Like all VE variants, ve32.exe is compiled for the 1MB real-mode 8086 address space and is thus limited to conventional non-UMB memory only.

Starting VE:

To start VE, simply type "ve" or "ve filename" (or “ve88”or "ve32") at the DOS prompt and you are off and running. The included Quick Start Guide / "Quick Start" page of this User Manual will help you to be immediately productive with VE.

Please remember that this User Manual contains of a wealth of useful information that will allow you to get the most value from VE. If you like what you see when you try out VE, please consider taking the time to read through this manual.

2.7 Acknowledgments

CampbellWare would like to acknowledge the contributions of:

1. Intel Corporation – Intel's Intellec and iRMX86 HI **Alter** text editor provided the initial inspiration for VE's two line, horizontally stacked user interface model. **Alter** also provided the inspiration for the names of several of VE 1.0's initial commands.
2. GB Boyd – in 1984, GB Boyd directly authored and contributed to VE 1.x a library of low-level tab and white space handling routines that are still incorporated into VE today, almost unchanged after all these years. Excellence stands the test of time!

3.0 VE Quick Start Guide

The following is a one page quick start guide to help you be productive with VE right away. This page is also available as the separate one page Quick Start Guide, included in the distribution .zip file.

Start VE to Edit One or More Files
At the DOS prompt, type “ve” or “ve filename” or “ve filename1 filename2 … filename”n”.

Text Entry
Simply type the text to be entered. VE places typed text into the file. Unlike editors such as vi/elvis, VE defaults to being in text entry mode vs. being in command mode.

Copy/Paste a Line of Text
Place the cursor anywhere on the line to be copied and type CTL-c. Move to where you want to paste the line and type CTL-v. Note that VE pastes above the line the cursor is on when CTL-v is typed.

Delete a Line of Text (CTL-x, F9)
Place the cursor anywhere in the line of interest and type CTL-x or F9.

Move a Line of Text (CTL-x, then CTL-v)
Place the cursor anywhere in the line to be moved and type CTL-x. This deletes it from its initial location. Go to the intended destination of the move and type CTL-v. This pastes it there.

Find Text (ESC-f, F5), Replace Text (ESC-r, F6)
Type ESC-f or F5 to find text in the file and jump to it. Type ESC-r or F6 to find text and then replace it with different text.

Repeat Last Command (F1, ESC-a)
Type F1 or ESC-a to repeat the last command. This is very useful with many commands, including the Find and Replace commands, where repeated use of F1 will sequentially find/replace successive occurrences. Alternately, to replace all occurrences at once, enter the '*' repetition count before the replace command (ESC * ESC-r, or ESC * F6).

Undo Changes (ESC-u, CTL-z)
Type ESC-u or CTL-z to undo changes made to a file. Sequential use of Undo will undo successive changes up until no changes remain. Note the VE does not support a Redo command.

Move Back and Forth Between Open Files (ALT-+, ALT–)
Type ALT-Keypad+ (press ALT and “Keypad+”) to switch to the **next** open file. Type ALT-Keypad- (press ALT and “Keypad-”) to switch to the **previous** open file. On the original IBM PC or on the IBM PC XT, use “CTL-]” and “CTL-[“ respectively instead.

Save Your Work and Continue Editing (CTL-s, F11)
Type CTL-s or F11 to update the current file on disk, and then continue editing it. This is equivalent to the File, Save sequence of most GUI editors. If it is a new file, VE will prompt for a file name before writing it to disk.

Save Your Work and Quit VE (ESC-q, F12)
Type ESC-q or F12 to run the Quit command. This command presents a menu allowing you to save your file and then quit VE, discard the current file and quit VE or discard the current file and resume editing of another open file (if more than one file is open).

4.0 Running VE

Run VE with one of five command forms:

1. ve
   This form starts VE and opens an empty new unnamed file.

2. ve -b
   This form starts VE in directory browsing mode. VE presents a full screen directory browser and allows interactive selection of the file to edit.

3. ve filename
   This form starts VE and opens the named file for editing

3. ve filename_1 filename_2 ... filename_n
   This form starts VE and opens all of the files provided on the command line for editing.

   Wildcards may always be used (for example, “ve *.txt”) with **ve32.exe.** They may also be used with ve.exe if your command interpreter expands these as they are handed to programs. Note that command.com and NDOS.com do not perform this function. All identified files will be opened for editing, up to a maximum of 256 files.

   You may use the “ALT +”, “ALT -” keys, or the Edit (ESC-e) “+” and “-” subcommands to cycle through the open files. Alternately, the Goto File command (ESC-g, “ALT *” or “CTL-*”) allows you to select a specific file from a full screen list of all open files. See the documentation for the Goto File command, below, for more details.

4. ve filename -l line_number
   This form starts VE, opens the supplied filename and moves to the indicated line number in that file. This is extremely useful for moving directly to the line number indicated in a compiler error message, or going directly to an area of a file that you are actively working at.

5. ve -h (or ve –-help, or ve/h)
   This form starts VE and displays the help screens. When the user exits the help panel, VE exits back to the command line.

At startup, VE will look for a file named $HOME\ve.opt (or $HOME\ve32.opt for ve32.exe). If present, VE will load this file and set its options from the file contents. This file may be created from within VE by setting program options as desired, and then using the Options menu Save subcommand to save the current option set. VE saves the option set to the file $HOME\ve.opt ($HOME\ve32.opt for ve32.exe). This functionality is only operative if the HOME environment variable is set.

5.0 Multiple Screen Resolution Support

Most EGA and VGA BIOS implementations are capable of placing the display into different text mode resolutions in addition to the default 80x25. VE fully supports this and can be run at a variety of different screen resolutions as detailed below.

5.1 Running VE at 80x50 (VGA Only)

Almost all VGA BIOS implementations are capable of placing the display into 80x50 text resolution, which is 80 columns wide by 50 lines deep. This is twice the default VGA text mode resolution of 80x25, and so provides twice as many lines of textual information per display page as the default. If the display in use is of a sufficient physical size to make this a comfortable editing experience, this can be quite useful.

VE supports this, providing an additional startup option that switches the display to 80x50 as VE starts up and then returns it to the default 80x25 when VE exits. This option is “-s 80x50” and it may be placed anywhere on the command line that initiates a VE session. For example:

ve -s 80x50 filename

will begin editing of “filename” after switching the display to 80x50. When VE exits, it returns to the resolution that was set to at the time VE was started. This will typically be the default 80x25 resolution.

5.2 Running VE at 80x43 (EGA and VGA)

VE also supports a startup option that switches the display to the less commonly used 80x43 text resolution. That option is “-s 80x43”. Like “-s 80x50” it may be placed anywhere on the command line that initiates a VE session. For example:

ve -s 80x43 filename

will begin editing “filename” after switching the display to 80x43. When VE exits, it returns to the resolution that was set to at the time VE was started. This will typically be the default 80x25 resolution.

5.3 Running VE at 80x25 (MDA, CGA, EGA, VGA)

By default, VE starts at a default 80x25 resolution for all of MDA, Hercules, CGA, EGA and VGA, unless the display was already at a different resolution. See the other paragraphs in this section for more details.

However, for completeness VE also supports a “-s 80x25” start up option, to provide for the case where the display has been placed into any text mode that is not 80x25 by an external DOS command but the user wishes to run VE at 80x25. In this case VE can be started with the “-s 80x25” option and as it starts, VE will change the resolution of the display to 80x25, operate in that mode until the user exits, and then return to whatever resolution was set at the time VE was started.

5.4 Running VE From Non 80x25 Resolutions

There are multiple software packages available for DOS that will shift the resolution of the display to not just 80x50 but many other resolutions as well. VE understands this and queries the BIOS at startup to determine the text resolution the screen is set to. If VE has not been started with an overriding “-s” startup option, VE will run at whatever resolution the screen is set to when it is started.

6.0 Graphics Types and VE

6.1 MDA, Hercules, CGA, EGA and VGA

When IBM introduced the first IBM PC in 1981, it provided two graphics options: MDA (Monochrome Display Adapter) and CGA (Color Graphics Adapter). In 1982, the third party Hercules graphics adapter was debuted in the market. Hercules offered sharper, clearer text and a superior graphics capability vs. CGA. As a result, Hercules was an instant sales success and for several years was the de facto standard graphics adapter for IBM PCs, PC XTs and clones.

Shortly after the PC AT was launched in 1984, EGA (Enhanced Graphics Adapter) became an available option. EGA improved on MDA, CGA and Hercules, offering more text modes and better color and graphics capabilities than any of its predecessors. In 1987, accompanying the launch of the IBM PS/2 line, IBM introduced VGA (Video Graphics Array). VGA again improved on the capabilities of its predecessor (EGA) and has become THE standard baseline graphics capability for all PCs sold, even to this day.

6.2 VE Graphics Adapter Support

VE supports all 5 of these graphics adapters, auto-detecting the deployed adapter at VE startup time. You can see what graphics type VE has auto-detected by using the Status command (ESC-s). Near the lower right corner of the Status page that is presented, you will see the detected graphics type. BTW, VE also auto-detects the CPU type, and the detected CPU is also presented by the Status command in the same general area of the Status screen.

VE understands the capabilities and limitations of each of these graphics types, and will prevent users from attempting actions that are not supported by their current graphics capability. For example, VE will not allow attempts to change its color schema when running on MDA or Hercules – neither of these adapters by default supports text mode color (although determined programmers can achieve some limited color on MDA and Hercules with clever coding techniques). Similarly, VE will not allow the “-s 80x50” option when running an any graphics type except VGA, since only this type supports that text resolution.

Hercules deserves a specific mention. Hercules supports both a text mode and a graphics mode. VE makes use only of its text mode. In text mode, Hercules fully emulates MDA, and thus can be treated as if it was an MDA adapter. This is what VE does. There is no Hercules-specific support; instead VE treats Hercules as if it was MDA.

6.3 VE “-g” Startup Option

As mentioned above, at launch time, VE auto-detects the graphics type deployed in the machine it is running on. Although there have been no reported cases of this, it is possible that VE might not correctly auto-detect the deployed graphics type and thus limit some operations that it would otherwise allow.

To provide for this, VE 4.4 supports a new startup option: “-g”. This option allows users to manually select their graphics type. Their selection overrides the auto-detected selection and VE makes use of the user-entered graphics type for the remainder of the editing session.

The “-g” startup option is entered on the VE command line as shown below:

ve [-g MDA | CGA | EGA | VGA] filename

An example is:

ve -g VGA filename

This example launches VE to edit the file “filename”, specifying that the graphics type is VGA. No matter what graphics type VE auto-detects, it will use VGA instead.

The ability to manually select the graphics type can be valuable, but it should be used with caution. For example, if you override the detected graphics type to MDA, but you are actually on CGA, the monitor will go blank and nothing else will happen visibly. VE is running just fine, but it is “displaying” via the MDA text buffer address, which is different from the CGA, EGA and VGA text buffer address. Caveat emptor!

7.0 VE User Interface

7.1 Entering Text

VE starts and runs in text entry mode. VE is normally in INSERT mode (characters typed are inserted into the file) but can also be placed in OVERWRITE mode (characters typed overwrite existing characters in the file). VE will scroll the display window to the right if the line being typed exceeds the width of the physical display or display window, up to the maximum supported line size (384 characters in the current version of VE).

VE always displays it's current mode (Insert, Overwrite or Command) in the upper right corner of the display.

7.2 Entering VE Commands

VE commands are entered by typing any one of:

- ESC followed by a command letter (type ESC first, then type the command letter)
- ALT plus a command letter (hold down ALT while typing the command letter)
- CTL plus a command letter (hold down CTL while typing the command letter

A repetition count can be used in addition to any of these forms, by preceding them with ESC- “n”, where “n” is the desired number of repetitions. For example, to paste the same line 5 times, type:

“ESC-5 CTL-v”

A convenience shortcut is available when entering commands using the ESC-command form. In this case, while you CAN enter “ESC-n ESC-command”, VE recognizes “ESC n command” and executes the command “n” times.

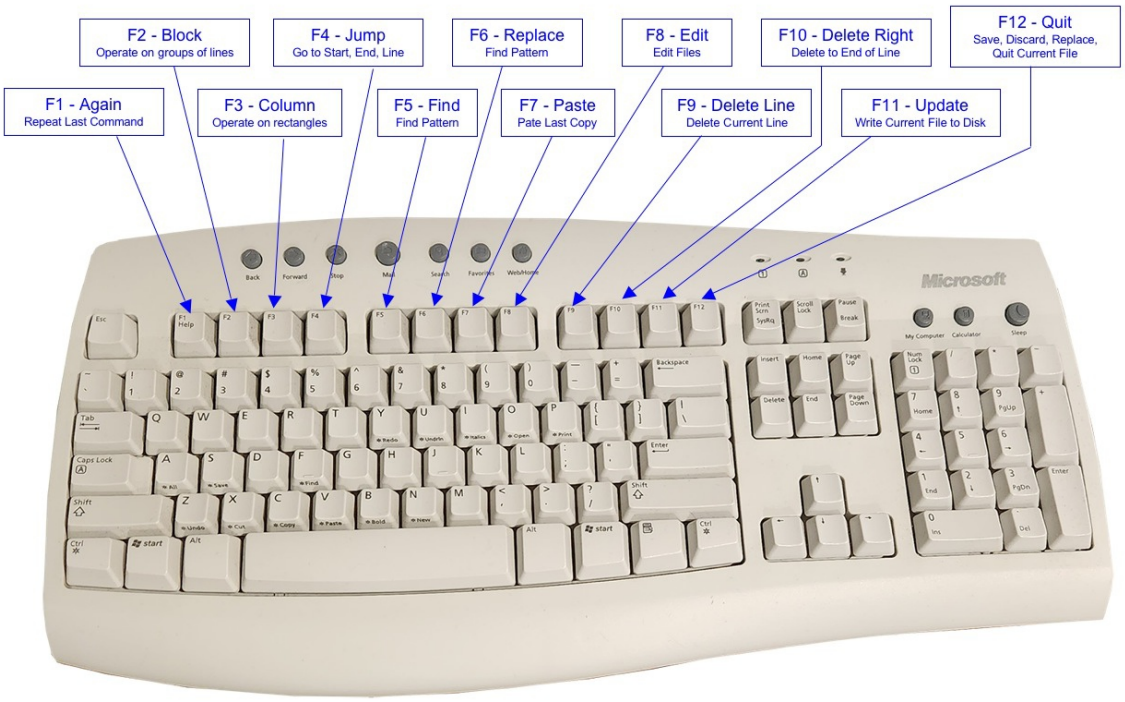
The full set of available commands is listed in the VE Help Panel, available via ESC-h or ALT-h, and in Appendix A of this manual. VE commands **are** case sensitive. For example, ESC-a is the “Again” command, while ESC-A is the “About VE” command.

The most common VE commands are bound to the PC keyboard function keys (F1 through F12), but **every** VE command can be executed via the ESC, ALT and CTL forms shown above, which means that VE can be used in any computing environment, including dial up, putty and Telnet, where function keys may not be understood or available.

VE **always** returns to text entry mode after executing a command. This makes VE far safer for the uninitiated than text editors like vi/elvis, where typing characters without first entering the INSERT or REPLACE commands can cause catastrophic results, as each typed character is interpreted as a command and executed.

7.3 Function Keys (F1-F12)

The PC Enhanced Keyboard function keys F1-F12 are bound to a set of frequently used VE commands, providing a fast and convenient shortcut to those commands. These are:



- F1 Again: repeat the last command
- F2 Block: select a group of continuous lines, copy, delete, append, paste, more
- F3 Column/Rectangle: select a rectangular area, copy, delete, append, paste, more
- F4 Jump: jump to start of file, end of file, line number
- F5 Find: enter a text pattern, search for it
- F6 Replace/Substitute: enter a text pattern and one to replace it. Search, replace.
- F7 Paste/Get: paste the last item copied (line, block, rectangle)
- F8 Edit: select files for editing
- F9 Delete Line: kill the current line
- F10 Delete Right: delete all text to the right of the cursor on the current line
- F11 Update/Save: write the current file to disk, resume editing
- F12 Quit: save/discard current file, replace current file with new file, quit VE, more

The command bound to each function key may be executed by simply pressing the associated function key. Appendix A presents a full listing of **all** command key bindings.

7.4 Exiting From Menus, Prompts, Etc.

The [ENTER] key will terminate any menu that VE is presenting, and will complete any input that VE is prompted for. This is also true for exiting from command mode. If you have entered ESC as the start of a command, it is possible to exit without running any command by simply pressing ENTER.

7.5 Command Repetition Counts

As mentioned above, VE's general command syntax is:

- ***ESC (optional repetition count) command_letter***
- ***(optional ESC repetition count) ALT command_letter***
- ***(optional ESC repetition count) CTL command letter***

Any VE command can include a repetition count. The command will be executed as many times as indicated. The special repetition count '*' means to repeat infinitely – this is useful with the Replace command, where it in essence means replace all occurrences, but should be used with extreme caution elsewhere. For example, executing a macro infinitely can have unexpected and unfortunate results depending on the content of the macro.

To enter a repetition count prior to a command, press ESC to begin command entry, and then the count itself. When the desired count has been entered (the count is echo'd on the status line and may be edited via the Backspace/Rubout key), simply enter the intended command, and that command will be sequentially executed the number of times indicated by the repetition count that has been entered. If the command has an extra keyboard “shortcut”, such as a Function key, this may be used instead of the command key.

7.6 Repeating Commands Sequentially

The VE **Again** command (ESC-a, F1) can be used to repeat most commands that change the file (such as delete char, delete line, replace text, line/column cut, copy, paste, etc.), the Find and Replace commands and selected others. The Again command repeats the last command including any repetition count that was initially entered with it. The Again command can be used repeatedly to achieve as many sequential executions of a command as is desired. Commands that do no change the file, such as cursor movement commands, Page Up, Page Down and more, do not support being repeated via the Again command.

7.7 Mouse Support

VE was designed to be a keyboard-driven text editor and operates primarily in that paradigm. However, some very useful mouse capability is supported. When VE starts, the mouse pointer is not visible. To activate the mouse and make the mouse pointer visible either left click the mouse or enter the ALT-m (Mouse) command. This same command can be used to hide the mouse pointer again when not being used.

When the mouse has been activated, there are several things that can be done with it:

- Position the cursor to any text area by left clicking there
- Select a word in the Column/Rectangle command by double clicking it
- Select a rectangular region in the Column/Rectangle command by left clicking on the upper left of the region, following by right clicking on the lower right of the region (drag and release is also supported for this).
- Select a file from any full screen file dialog by left clicking or double clicking on it.

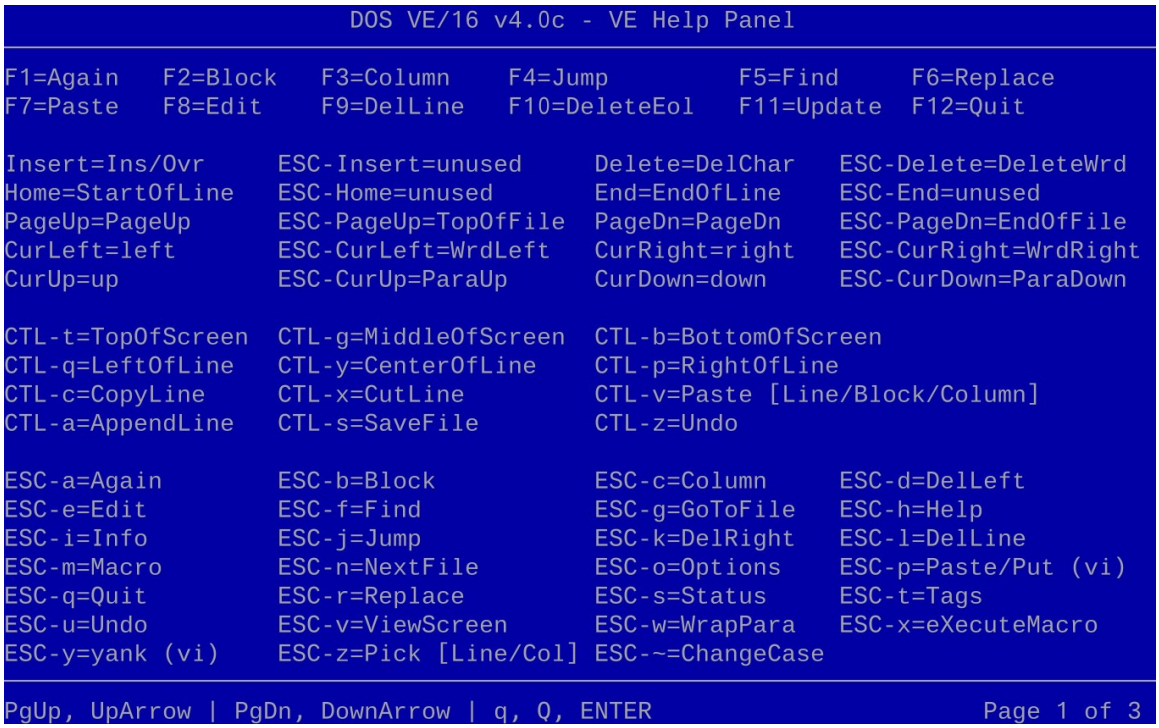
7.8 Getting Help

This document is the primary source of detailed information about the VE editor. However, VE provides two other help sources that can be accessed from within VE while it is running. These are:

The VE Help command (ESC-h/Alt-h)

The Help command presents the VE Help Panel, which provides a list of all available commands and their command keys. This information is presented as a multi-page full screen panel. At the bottom of each panel screen VE presents a navigation guide, allowing users to page back and forth through the panel and when done, leave it and return to their editing.

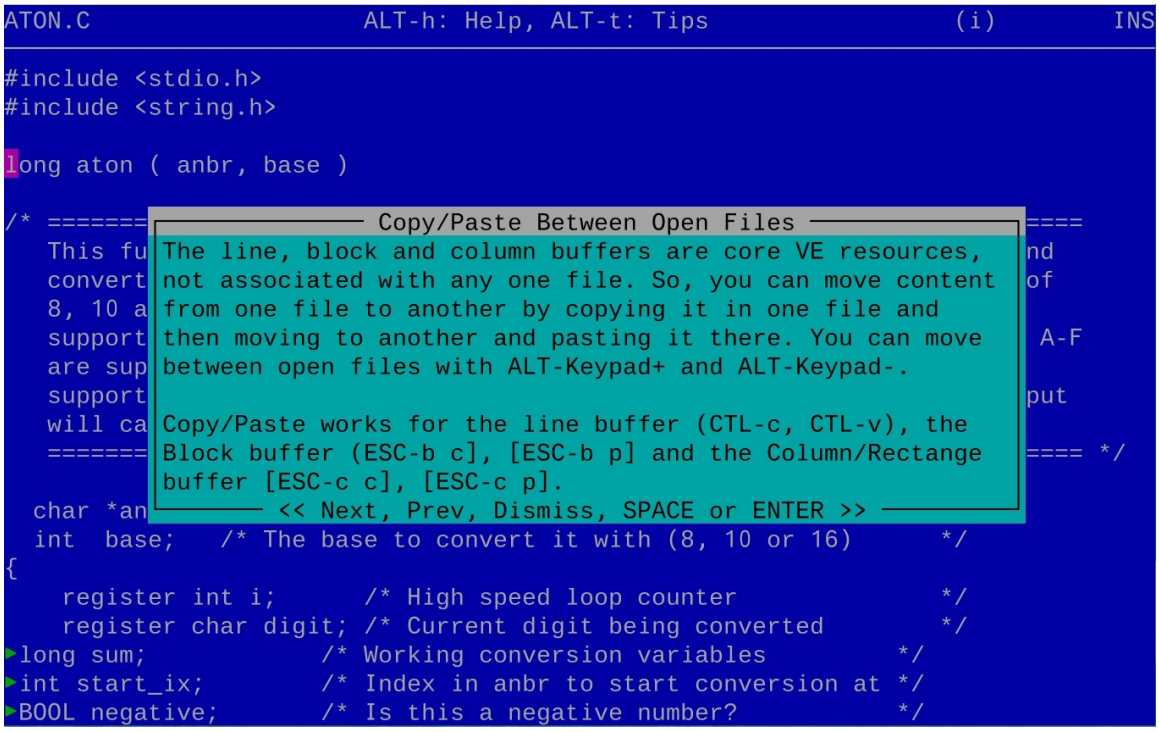
The screenshot below shows the initial page of the help panel.



The VE Tips command (ALT-t)

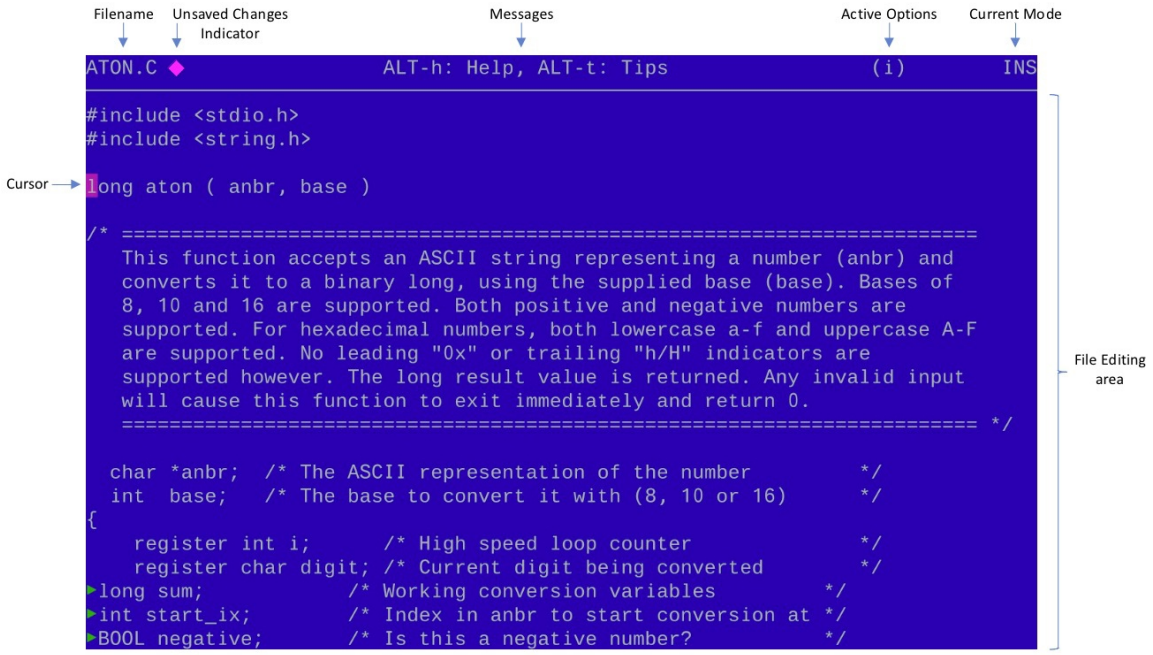
The Tips command provides on-demand tips. Each time it is executed, it presents one randomly selected VE tip in a pop-up window. The file C:\ve.tps must be present in order for this command to provide the intended tips. At the bottom of the tip pop-up, VE presents a tip navigation guide, which allows users to view the next or preceding tip, and when done, dismiss the tip window and return to their editing.

The screenshot below shows a typical tip being displayed.



8.0 VE Screen Layout and Interaction Model

8.1 VE Screen Shot




```
#include <stdio.h>
#include <string.h>

long aton ( anbr, base )

/* ========================================================================
   This function accepts an ASCII string representing a number (anbr) and
   converts it to a binary long, using the supplied base (base). Bases of
   8, 10 and 16 are supported. Both positive and negative numbers are
   supported. For hexadecimal numbers, both lowercase a-f and uppercase A-F
   are supported. No leading "0x" or trailing "h/H" indicators are
   supported however. The long result value is returned. Any invalid input
   will cause this function to exit immediately and return 0.
   ======================================================================== */

  char *anbr;  /* The ASCII representation of the number         */
  int  base;   /* The base to convert it with (8, 10 or 16)      */
{
    register int i;      /* High speed loop counter              */
    register char digit; /* Current digit being converted        */
long sum;            /* Working conversion variables         */
int start_ix;        /* Index in anbr to start conversion at */
BOOL negative;       /* Is this a negative number?           */
```

8.2 VE Screen Layout Notes

The VE screen layout consists of the following logical areas:

- File Editing Area: this is the majority of the screen, and is the area where the file content is displayed and can be edited. This page may be scrolled up, down, left and right to display all of the file content.
- Status Line: this is the top line of the screen, and is the area where VE presents information about the current file, any relevant messages/information, the currently active options and VE's current mode.
- Filename Field: this is at the leftmost edge of the status line, and displays the name of the file currently being edited. If the filename is longer than the field width, it may be truncated in this field, with ellipses ("...") showing as the first three characters.
- Unsaved Changes Indicator: this is a colored diamond shape displayed immediately to the right of the current file name if changes have been made to that file since it was opened, or since its last update to disk. Once a file with unsaved changes is updated to disk (CTL-s, ESC-U, F11), this indicator goes out until the next unsaved change is made.
- Status Line **Messages** Field. This is in the middle of the status line, and is the area where VE presents any messages or information that it wishes the user to see. The majority of VE commands use this area to present status information reflecting what they have done. For example, the Replace command reports the number of substitutions made in this area.

- Status Line **Options** Field. This is to the right of the Messages field. VE uses this area to display indicators reflecting the status of options the user has enabled. The available indicators are:
  - "i" - indicates that auto indent is active
  - "w" - indicates that real-time text wrapping is active
  - "p" - indicates that Power Typing (an historical form of text wrapping) is active
  - "m" - indicates that macro recording is active
- Scroll Indicator. "<-" - indicates that the display page is scrolled to the right
- Status Line **Mode** Field. On the right hand edge of the status line, this is where VE presents its current mode, which is one of Insert (INS), Overwrite (OVR) or Command (CMD)

8.3 Status Line Command Interaction Model

With the exception of full screen file selection dialogs, all command dialogs with VE occur on the status line. VE presents menus and provides command feedback from menu selections made in this area. Where more detailed feedback from a command is needed, VE will temporarily deepen the status line area to up to three lines deep and utilize the extra lines as needed. When the interaction is complete, VE restores the status line to its original single line depth.

Since the status line is only one line "tall", and VE may need to present nested menus (if for example a selected command has a menu, which itself has a menu), VE treats the status line like a push down stack. When a VE command needs to present a menu, it conceptually pushes the current status line onto an internal status line stack, uses the status line as it wishes, and then pops the previous status line off the internal status line stack when it is done, restoring the previous status line.

Since ENTER will terminate almost any menu that VE is presenting, to move "up" through stacked menus (i.e. to get back to the previous status line menu) simply press ENTER repeatedly until you return to the menu of interest, or back to editing the file.

8.4 Status Line Response Line Editing

Many VE commands present information on the status line and request user input. A common example of this is the Find command, which prompts the user for a string to find. Another common example is the [Jump, Line] command, which prompts for a line number to jump to. In all cases where a user is prompted to enter text, full line editing support is provided while that text is being entered. The following line edit keys are supported:

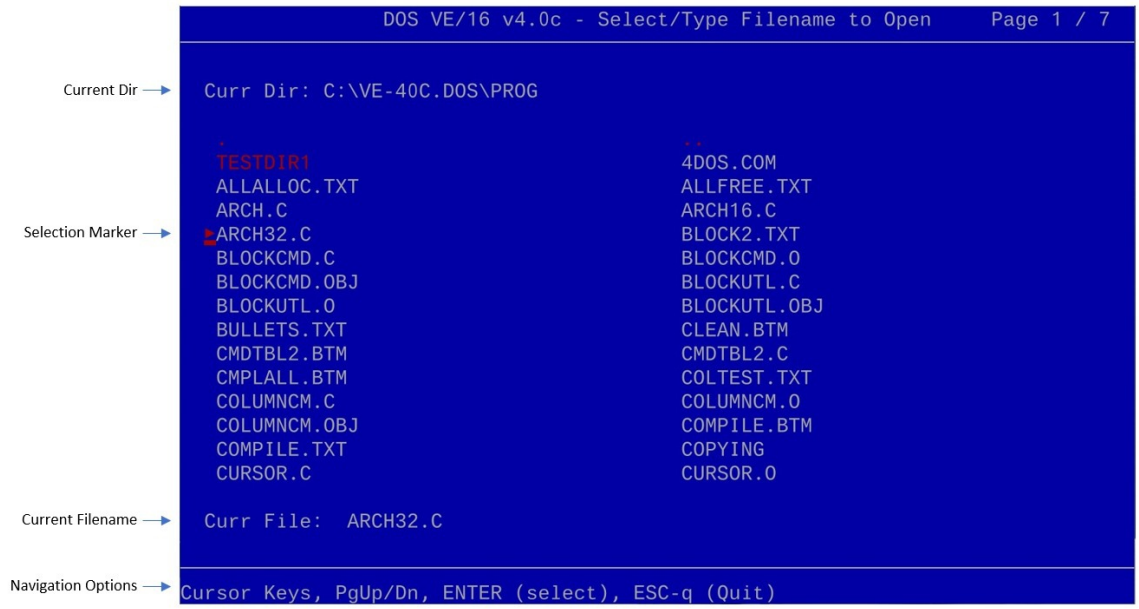
- Rubout/Backspace – erases the most recently entered character
- Delete – deletes the character under the cursor
- Home – moves to the start of entered response
- End – moves to the end of the entered response
- Cursor Left – moves the cursor left one position
- Cursor Right – moves the cursor right one position
- F10 or ESC-k – deletes all response characters to the right of the cursor
- ESC-d – deletes all response characters to the left of the cursor

Delete Right (F10 or ESC-k) deserves special mention with respect to the VE Find command. A common use of the Delete Right capability occurs with this command, which remembers the last find string that was entered, and preloads the presented find string with the last one used. To enter a new string instead of reusing the previous string, the fastest approach is to enter F10/ESC-k, which clears the previous string, and then enter the desired new string.

8.5 VE Open File Dialog

A number of VE commands allow users to select or specify filenames for reading, writing or opening. All such commands use a common VE Open File Dialog, which includes an interactive file system browser. This dialog allows users to select a file from the current directory, or browse the file system to a new directory and select a file from there or alternately, directly type a partial or complete filename.

A screen shot of the Open File Dialog appears below:



8.5.1 Selecting Files Interactively

The Open File Dialog (and the Goto File dialog) presents a full page listing of the contents of the directory from which VE was started. Cursor left, right, up and down keys may be used to move the file selection marker to the file of interest, which may then be selected by pressing ENTER. If the directory listing exceeds one display page, the PgUp and PgDn keys may be used to move to the next/previous page of the listing as desired. As an alternate to using the PgUp or PgDn keys, the cursor up and cursor down keys may be used. When they reach the top or bottom of the displayed files, if there is another page, they will wrap to that page. Note that the cursor keys will wrap from the left to the right, and visa versa, and will wrap from the top to the bottom and visa versa. In similar fashion, the PgUp and PgDn keys will wrap from the first page to the last, and visa versa.

If the file of interest is not in the directory that is displayed, the Open File Dialog may be used to browse the file system until the directory of interest is reached. To do this, simply select any directory entry and press ENTER (or double click the directory entry with the mouse). This will open the selected directory and present its contents. This process can be repeated until the directory of interest is reached.

To move up to the parent directory, simply select the “..” entry at the top of the file list, or use the ALT-PgUp keystroke (on PC and PC XT, use CTL-PgUp instead). Any of these will move you up to the parent directory.

If the Open File Dialog is used to select a file that VE already has open, VE will simply switch to that file, rather than opening a second copy of it.

Users may cancel out of the Open File Dialog at any time without selecting a directory and file by pressing ESC-q. F12 achieves the same result.

The cursor movement and PgUp, PgDn functions may also be selected by their keyboard command keys. This may be of some value in environments like putty and Telnet. Specifically, the following alternate forms are available while in the Open File Dialog:

- Cursor Up - CTL-u, ESC-^
- Cursor Down - CTL-d, ESC-&
- Cursor Left - CTL-l, ESC-,
- Cursor Right - CTL-r, ESC-.
- Page Up - CTL-o, ESC-@
- Page Down - CTL-n, ESC-F

8.5.2 Selecting Files Using the Mouse

The Open File Dialog (and the Goto File dialog) supports file selection via the mouse as well as the keyboard. Single clicking the mouse pointer on a directory entry will move the selection carat to that entry. Double clicking on a directory entry will select the file that is double clicked and complete the dialog, as if the selection carat had been moved to the file of interest and ENTER had been pressed.

8.5.3 Selecting Files by Typing Their Name

As an alternate to interactively selecting a directory and file of interest, **a partial or full path name may be directly typed**. It will be echoed in the Curr File field, complete with full line editing support. Partial filenames are assumed be relative to the currently displayed directory, while full pathnames (those starting with "\" or with a drive letter) are assumed to be absolute paths. Note that relative pathnames of the form "..\..\directory_name\filename" are fully supported.

8.5.4 Selecting Files Using Filename Auto-Completion

The VE Open File dialog (and the VE Goto File dialog) support filename auto-completion. To use auto-completion, type a partial filename and then press the TAB key. VE will present the first filename whose initial characters match the partial filename typed. If this is not the file of interest, press TAB again. VE will present the next matching file. Carry on in this manner until the desired filename is presented and then press ENTER to select it.

When VE has presented the last matching filename, if TAB is pressed again, VE will beep once. This indicates the end of the set of matching files. If TAB is pressed again after this beep, VE will recycle to the start of the list of matching filenames and present the entire list again each time TAB is pressed. This allows users to cycle through the list again if they inadvertently pressed TAB one time too many and tabbed past their intended file.

Auto-completion is supported for both filenames and directory names. If a partial directory name is auto-completed, VE will update the full screen display each time TAB is pressed, so that it shows the contents of the currently auto-completed directory name. Continue to press TAB until the desired directory name is presented and then press ENTER to select it. Carry on with further directory names until the directory of interest is achieved and then move into typing the desired file name. These too can be partial names with auto-completion.

Filename auto-completion supports three special characters that modify how it works. These are:

- “^” - indicates “starts with”. For example, typing “^hello” and then TAB will auto-complete matching all files whose file name starts with “hello”. This could match such things at “hello123.txt”, “hellojnk.dmp”, “hello.c” and so on.
- “**$**” - indicates “ends with”. For example, typing “txt$” and then TAB will auto-complete matching all files whose filename ends with “txt”. This could match “total.txt”, “original.txt”, “readme.txt” and so on.
- “**~**” - indicates “includes”. For example, typing “~del” and then TAB will auto-complete matching all files whose filename includes “del” anywhere in the name. This could match such things as “abdel.txt”, “delete.c”, “xyz.del” and so on.

Note that the characters used for “starts with” and “ends with” can be changes using the Options command Match subcommand. The character used for “includes” is fixed as “~” and cannot be changed.

9.0 A Few Words About...

9.1 A Few Words About Tab Handling

With its default settings, VE expands Tabs into the equivalent number of spaces as a file is read in. Tabs may be optionally re-inserted as a file is written back out. In general VE views tabs simply as a way of compressing a file's size on disk. Tabs are expanded when a file is read in, and may optionally be used to compress a file's size when the file is written back out.

This behavior can be changed via the [Options, Tabs] menu path (ESC-o, t), which allows both input and output tab translation to be enabled or disabled. If Input Tab Translation is disabled, VE will represent TABs using a (typically green colored) carat symbol. Similarly, for Tab characters entered into a file as text, VE displays those characters using the same (typically green colored) carat symbol. When Input Tab Translation is enabled, VE will expand Tabs into the appropriate number of spaces as a file is read in, and will also convert any TABs typed as text to the number of spaces needed to achieve the next tab stop.

The carat symbol used to display the Tab character in files is typically green in color, but VE will dynamically change this to another color in the event that the user selects a color scheme where the foreground or background color is set to green.

When expanding tabs on file input, VE uses a default tab stop of 4. If this tab stop causes the visual alignment of text in a file to be incorrect, determine the correct tab stop setting for that file. Then change VE's tabstop setting via the [Options, Tabs] menu path (ESC-o, t), discard the file and re-open. Alignment should now be correct. Note that when you write the file back out, it will not contain tabs unless you enable use of tabs on output via the [Options, Tabs] menu path (ESC-o, t). Input and Output tab translation are disabled by default, but may be changed using the Options command Tab subcommand.

Finally, an application note. Users of various “make” utilities should note that many such utilities REQUIRE their makefile input file to contain tabs. When editing such make files, users MUST disable Input Tab Translation. If this is not done, the saved makefile will not be acceptable to the “make” utility.

9.2 A Few Words About Text File Formats

VE has been ported to both Linux and Mac OS X Terminal. VE therefore lives in a mixed DOS, Windows, Linux and MacOSX world. Many machines that VE will run on may contain multiple operating systems. A well known difference between DOS/Windows and Linux/MacOSX is the format in which they store text files. Unix/Linux/MacOSX uses a single Line Feed character (0x0A) as the separator between text lines in text files. DOS/Windows uses the combination of carriage return (0x0D) and Line Feed (0x0A).

To ensure that both DOS/Windows and Linux/MacOSX files can be read and written properly, VE supports both formats, determining on the fly as a file is opened which text format it is using. VE then uses the same format when writing that file out. For new files created with DOS VE, the default format is the DOS one, but this can be overridden, via the [Options, Fileformat] menu path (ESC-o, f). This command allows you to set the text file output format to either DOS or Linux. Hence, VE running from DOS can be used to create new text files that will be stored on a Linux-accessible DOS/Windows partition of the same machine, in a manner that will allow Linux text editors to open them normally.

The text file format of an open file may be checked at any time. When VE opens a file, it displays information about the file on the Status line. This includes the number of lines and characters, and it also directly shows the file format that VE has determined that the file is using. At any time during editing of a file, you may check the file format by using the Status command (ESC-s) or the Goto File command (ESC-g), both of which display full statistics about the current editing session, including the file format of the currently edited file.

10.0 Entering Text

10.1 Entering Text

To enter text in VE, simply begin typing. VE defaults to being in text insertion mode. VE supports two text insertion modes: INSERT mode (new text is added to the file) and OVERWRITE mode (new text overwrites existing text). On startup, VE is always in INSERT mode. All characters typed will be entered into the file content.

As mentioned earlier in this document, this simple fact (VE defaults to being in text entry mode) makes VE a much safer choice for many users than text editors like vi/elvis, where typing characters without first entering INSERT or REPLACE mode can cause catastrophic results (each typed character is interpreted as a command and executed!).

VE has another very helpful behavior while entering text. In many editors, if you wish to place additional text beyond the end of the last character on a line, you have to get into text input mode and then space over to where you want the new text to begin before you can begin typing it. This can be time-consuming and annoying. VE is different. VE allows you to place your cursor anywhere on screen, whether it is part of or outside of currently entered text, and start typing right away. Any distance between the current end of line and the newly typed characters will be auto-filled with spaces.

This may not sound like much of an improvement (just using cursor keys vs. Space key) but consider the case where there is a line above or a line below that has text in roughly the same column that you want to add text to on the current line. You can simply use large scale cursor motion commands like End to move to the desired column, cursor up or down to the line of interest and start typing. This can be an incredible time saver. There are many other ways to achieve quick and large scale cursor motion which multiply the value of this simple feature.

VE will automatically scroll the display to the right if the text being entered exceeds the width of the display. The current version of VE supports a maximum line width of 384 characters.

To toggle between INSERT and OVERWRITE mode, press the INSERT key on the keyboard, or enter ESC-I (note that the “I” is an upper case command letter). VE displays it's INSERT/OVERWRITE mode in the upper right corner of the screen as “INS” and “OVR” respectively. VE will toggle between INSERT and OVERWRITE as the keyboard INSERT key is repeatedly pressed (or ESC-I is repeatedly entered).

Tabs, when entered as text into a file, will be represented by a (typically green) carat symbol, if the Input Tab Translation option is enabled, or will be expanded into the correct number of spaces to reach the next tab stop, if Input Tab Translation is disabled.

10.2 Breaking and Joining Lines

Any given line can be broken into two lines by placing the cursor at the point where the line break is desired, and pressing the ENTER key.

Two lines may be joined together by moving to the end of the upper of the two lines (place the cursor on the line of interest and press the END key) and pressing the DELETE key. Alternately, two lines may be joined by pressing Backspace/Rubout while in the first column of a line. That line will be joined with the line above it.

11.0 Formatting Text

VE supports both on-the-fly and after-the-fact text wrapping, which can be very useful when composing text documents such as letters, program documentation etc.

11.1 On-The-Fly Text Wrapping

On-the-fly text wrapping is text wrapping that occurs as text is typed, without the need of manually pressing ENTER. VE supports this mode of text input. Text can be entered and/or deleted continuously, and VE will format the lines and insert line breaks as needed.

To enable on-the-fly text wrapping, go to the Options menu, Wrap submenu (ESC-o, w) and select the "Enable" menu item. A "w" indicator will be displayed in the options area of the status line to indicate that on-the-fly text wrapping is active. Prior to enabling wrapping, the "Left" and "Right" menu items may be used to adjust the left and right margins within which VE will wrap text. Alternately, VE's defaults of left margin at -1 and right margin at column 72 may be used, in which case no change is needed. A left margin setting of "-1" instructs VE to determine the correct left margin to use automatically, based on the current level of indentation of the paragraph being typed in.

When On-The-Fly Text Wrapping is first enabled, VE attempts to wrap the current paragraph as a starting point. Text may now be entered continuously, and VE will wrap that text as it is entered. Text entry can continue in this way until all desired text has been entered. At this point text wrapping may optionally be disabled via the [Options, Wrap, Disable] menu path (ESC-o, w, d).

While text wrapping is on, it is possible to move freely through the file and type at any place. VE will begin wrapping any line of text as soon as (a) at least one character is entered on that line, and (b) the line length exceeds the selected right margin.

Per the earlier section on the history of VE, it has a long and storied history, dating back to v1.x in 1984. In that initial version of VE, what is now called text wrapping was referred to as "power typing" In a nod to those early days, VE still activates text wrapping when the command keys associated with the Power Typing option are entered. Hence, text wrapping can be enabled via the Options menu, Power submenu (ESC-o, p). When enabled this way, a "p" option will be displayed in the Options area of the Status line vs. the usual "w" option. Same functionality, different command name and key sequence!

11.2 After-The-Fact Text Wrapping

VE also supports after-the-fact text wrapping. After-the-fact text wrapping refers to entering one or more lines of text without on-the-fly text wrapping active, and then wrapping that text one or more paragraphs at a time after text entry is complete.

To do this, simply position the cursor in any paragraph to be wrapped, and enter the VE Wrap command (ESC-w). VE will reformat the paragraph to make it fit between the selected left and right margins, and will optionally right justify each line, giving the text a clean looking uniform right edge. This may be repeated on as many paragraphs as desired. Right justification is on by default, but can be disabled via the [Options, Wrap, Justify] menu path (ESC-o, w, j). This selection toggles the state of right justification between "on" and "off".

After-The-Fact Text Wrapping supports a special feature for unique first line indentation. If the text wrapper notices that all lines of the paragraph to be wrapped have an identical indentation level except the first line, it will assume that this is intentional, and preserve the indentation level of the first line of the paragraph when wrapping. This allows for the wrapping of such things as bulleted lists and letter style paragraphs, where the first line has extra indentation vs. the remaining lines of the paragraph. As an example, consider a paragraph of the form:

```
o This is the first line of a bulleted list
  paragraph. The following
  lines will have the same
  indentation level. Only the first line is different.
```

VE's text wrapper will recognize this construct, and wrap it with the first line indented at the selected left margin, while the rest are indented relative to this.

The result of wrapping the above paragraph would be similar to the following:

o This is the first line of a bulleted list paragraph. The following lines
  will have the same indentation level. Only the first line is different.

Multiple paragraphs may be wrapped sequentially by repeating the Wrap command as many times as desired, either by directly re-entering the command, or using the VE “Again” command (ESC-a or F1). Yet another way to accomplish this is to enter a repetition count ahead of the Wrap command. VE will then sequentially wrap as many paragraphs as the repetition count entered. Finally, VE also supports wrapping multiple paragraphs (or even entire files) at once via the Block Wrap command. See the documentation for the Block commands, later in this document.

12.0 Deleting, Copying and Pasting Text

VE provides a rich set of capabilities for copying, moving and deleting text. VE can operate upon characters, words, lines, paragraphs and rectangles.

12.1 Deleting Text

VE can delete characters, words, paragraphs and both line-oriented and rectangle-oriented text areas. Each of these is described below.

12.1.1 Deleting Characters with the Delete Key

The most obvious way to delete a character is to position the cursor on the character of interest and press the keyboard Delete key (or the keypad Del key). This deletes the character immediately under the cursor and leaves the cursor positioned on the next character to the right of the one deleted. If this leaves the cursor at the end of a line of text and the Delete key is pressed again, it will delete the line end, which has the effect of joining the line the cursor is on and the line immediately below it.

12.1.2 Deleting Characters with the Backspace/Rubout Key

Characters can also be deleted using the Backspace/Rubout key (hereafter referred to as Backspace). Backspace is also bound to the CTL-h keystroke. To delete using the Rubout key, position the cursor immediately to the right of the character to be deleted and press Backspace/Rubout.

In Insert mode, the character to the left of the cursor will be deleted and the cursor will remain on the initial character it was placed on (but the line will shift one character to the left). If this leaves the cursor at the start of the line and Backspace is pressed again, it will delete the line end of the line immediately prior to it, which has the effect of joining those lines together.

In Overwrite mode, the Rubout command rubs out the intended characters by “erasing” them back to their original values. In essence, in Overwrite mode Rubout restores the characters that were overwritten and moves the cursor to the last character that has been restored. So, if in Overwrite mode the characters “xyz” are typed over with the characters “123”, using Rubout immediately after typing “123” will “erase” the “3” back to “z” and leave the cursor on “z”. Continued use of Rubout will erase the “2” back to “y” and the “1” back to “x”.

12.1.3 Deleting Multiple Characters with Delete, Backspace

Like all VE commands, Delete and Backspace can be preceded by a repetition count. Hence,

ESC 5 Delete

will delete the next 5 characters to the right. Similarly,

ESC 5 Backspace

will delete the 5 characters immediately to the left of the cursor.

12.1.4 Deleting All Characters to the Right, Left

VE supports the Delete Right command (ESC-k, ALT-k, F10), which will delete all characters from the cursor position to the end of the line. The equivalent Delete Left command (ESC-d) will delete all characters from the cursor position to the start of the line. Delete Right leaves the cursor at the end of the line. Delete Left leaves the cursor at the start of the line.

Note that like many VE command keystrokes, the Delete Left and Delete Right command letters are spatially laid out. The Delete Left keystroke (ESC-**d**) is on the left side of the keyboard, while the Delete Right keystroke (ESC-**k**) is on the right side of the keyboard. This sort of spatial command key layout is used quite often in VE to assist in remembering which keystrokes perform which functions.

12.1.5 Deleting Words

The Delete Word (ESC-Delete, ESC-Del, ALT-Delete, ESC-W) command deletes the word the cursor is on, if the cursor is within a word at the time of the command, or deletes the word immediately to the right of the cursor if the cursor is between words at the time the command is issued. Delete Word always deletes to the right; there is no form of Delete Word that operates to the left.

Like all VE commands, Delete Word can be preceded by a repetition count. Hence:

ESC 5 ESC-Delete

will delete the 5 words immediately to the right of the cursor. Note that “ESC 5 Delete” will delete the next 5 characters, not the next 5 words, since Delete and ESC-Delete are bound to separate commands.

12.1.6 Deleting Lines

The Delete Line command (ESC-l, ALT-l, F9) deletes the line the cursor is on and moves the cursor to the line immediately below, unless the line just deleted was the last line of the file, in which case VE moves to the line immediately above.

Delete Line can also be used to delete multiple lines, by preceding it with a repetition count. Hence:

ESC 5 F9

will delete five rows, starting with the one the cursor is on at the time of the command.

Lines can also be deleted using the GUI-standard CTL-x keystroke, but unlike Delete Line, this command first copies the line to be deleted to the internal VE Line Buffer and then performs the deletion. After this, the contents of the VE Line Buffer can be pasted at a different location in the file using the GUI-standard CTL-v keystroke. F7 and ESC-p will accomplish the same result. All three activate VE's Paste command.

12.1.7 Deleting Paragraphs

VE understands the concept of paragraphs and is able to jump forward and backward a paragraph at a time (or multiple paragraphs at a time). VE treats paragraphs as a type of block of lines, and thus paragraph deletion is accomplished using the line-oriented Block command. To delete a paragraph, start the Block command (ESC-b, F2) and select the paragraph of interest by using the Paragraph Up or Paragraph Down commands (ESC-[, ESC-]) to move the selection markers. Then use the Block command's Delete (d, CTL-x) subcommand to delete the paragraph(s). Note that this has the effect of copying the paragraph(s) into the Block Buffer. From there, it/they can be pasted anywhere else in the file using the Block command's Paste (p, CTL-v) subcommand or VE's global Paste command (CTL-v, ESC-p, F7). See the “Operating on Blocks of Lines” chapter later in this document for full details on the Block command.

12.1.8 Deleting Blocks of Lines

Blocks of lines can be deleted using the Block command. To do this, start the Block command (ESC-b, F2) and select the continuous set of lines of interest. Then use the Block command's Delete (d, CTL-x) subcommand to delete the entire block of lines. Note that this has the effect of copying that block of lines into the Block Buffer. From there, it can be pasted anywhere else in the file using the Block command's Paste (p, CTL-v) subcommand or VE's global Paste command (CTL-v, ESC-p, F7). See the “Operating on Blocks of Lines” chapter later in this document for full details on the Block command.

12.1.9 Deleting Portions of Lines

Portions of a line (for example, a single word or a small set of words) can be deleted using the Column command. To do this, start the Column command and then select the line portion of interest using the selection markers. In this case, move directly to column selection; there is no need for line selection since the rectangle of interest is in fact just one line deep. Once the line portion has been selected, use the Column command's Delete subcommand (d, CTL-x) to delete it from the file. This has the effect of copying the line portion into the Column Buffer and so it can be pasted elsewhere in the file if desired (using the Column commands Paste subcommand or VE's global Paste command). See the “Operating on Column Areas” chapter later in this document for full details on the Column command.

12.1.10 Deleting Rectangular Blocks of Text

Rectangular areas of text can be deleted using the Column command. To do this, start the Column command (ESC-c, F3) and select the rectangular area of interest by moving the selection markers first vertically (unless the area is just one line deep) and then horizontally. Once the rectangle is satisfactorily defined, use the Column command's Delete subcommand (d, CTL-x) to delete the defined rectangle from the file. Note that this has the effect of coping that rectangle into the Column Buffer. From there, the rectangle can be pasted at any arbitrary location in the file (using the Column commands Paste subcommand or VE's global Paste command). Note that rectangles can include spaces beyond the end of a line – VE autofills the blank areas with spaces in the copied rectangle (this is not done in the file itself). For full details on the Column command, see the later section of this document “Operating on Column Areas”.

12.2 Copying Text

VE can copy complete lines, portions of lines, blocks of lines and rectangular areas of text. Each of these is described below.

12.2.1 Copying Lines

Individual lines can be copied using the Copy Line command (CTL-c, ESC-y, ESC-C). This command copies the line the cursor is on into VE's Line Buffer. From there, the line can be pasted elsewhere in the file via VE's global Paste command (CTL-v, ESC-p, ALT-p, F7). As will be seen later in this document, individual lines can also be accumulated in the Line Buffer by using the paired Append Line (CTL-a) command, which copies the line the cursor is on and appends it to the current contents of the Line Buffer.

Multiple lines can be copied at once by preceding the CTL-c or CTL-a commands with a repetition count. Hence:

ESC 5 CTL-c

will copy the line the cursor is on, and the four lines after it, into the Line Buffer.

Note that the ESC-y binding for the Copy Line command is a nod to the vi text editor's "yank" command. ESC-y, ESC-p will perform the VE equivalent of vi's yank/put set of commands.

Copy single lines can also be accomplished by starting the Block command (ESC-b, F2) and then immediately using its Copy subcommand (c or CTL-c) without moving the selections markers. This will copy the single line the cursor is on, but into VE's Block Buffer instead of its Line Buffer.

Like lines copied via CTL-c, multiple individual lines can be accumulated in the Block Buffer by selecting the lines of interest one by one, and using the Block command's Append (a, CTL-a) subcommand. This subcommand will append the selected block to the existing contents of the Block Buffer. For full details, see the later section of this document "Operating on Blocks of Text".

12.2.2 Copying Portions of Lines

Portions of lines can be copied using the Column/Rectangle command. A primary reason for wanting to copy a portion of a line is to capture an instance of a long or difficult to type word (or short set of words) and then paste it everywhere else it is needed, vs. having to retype it each time.

The Column command can be used for this. A single word, or a portion of any single line, can be thought of as a rectangular area that is simply one line deep. Hence to capture/copy a single word, or a small set of words, use the Column command, select just the portion of the line that is of interest (move directly into column selection – no need for line selection in this case) and use the "c" (or CTL-c) subcommand to copy it into the Column Buffer. From there, it can be repeatedly pasted into as many places as desired using the Column command's Paste (p, CTL-v) subcommand. Note that simply using VE's global Paste command (CTL-v, ESC-p, F7) will accomplish the same result. For full details on the Column command, see the later section of this document "Operating on Column Areas".

12.2.3 Copying Blocks of Lines

Blocks of lines can be copied using the Block command. To do this, start the Block command (ESC-b, F2) and select the continuous set of lines of interest. Then use the Block command's Copy (c, CTL-c) subcommand to copy the entire block of lines into the Block Buffer. From there, the block of lines can be pasted anywhere else in the file using the Block command's Paste (p, CTL-v) subcommand or VE's global Paste command (CTL-v, ESC-p, F7).

Multiple blocks of lines can be accumulated in the Block Buffer using the Append (a, CTL-a) subcommand instead of the Copy subcommand. The Append subcommand appends the current block of lines to the existing contents of the Block Buffer. The resulting expanded block of lines can then be pasted elsewhere in the file using the Block command's Paste (p, CTL-v) subcommand or VE's global Paste command. For full details on the Block command, see the later section of this document "Operating on Blocks of Lines".

12.2.4 Copying Paragraphs

VE treats paragraphs as a block of lines delimited with one or more blank lines on each side. Accordingly, paragraphs can be copied using the Block command. When selecting the lines of interest for the block, use the “]” key to move the marker forward one paragrph, or “[“ to move the marker backward one paragraph. Continue to use these keys until the desired number of paragraphs have been selected.

Then use the Block command's Copy (c, CTL-c) subcommand to copy the entire set of paragraphs into the Block Buffer. From there, the paragraphs can be pasted anywhere else in the file using the Block command's Paste (p, CTL-v) subcommand or VE's global Paste command (CTL-v, ESC-p, F7).

Multiple paragraphs can be accumulated in the Block Buffer using the Append (a, CTL-a) subcommand instead of the Copy subcommand. The Append subcommand appends the current paragraphs to the existing contents of the Block Buffer. The resulting expanded set of paragraphs can then be pasted elsewhere in the file using the Block command's Paste (p, CTL-v) subcommand or VE's global Paste command. For full details on the Block command, see the later section of this document “Operating on Blocks of Lines”.

12.2.5 Copying Rectangular Areas

Rectangular areas of text can be copied using the Column command. To do this, start the Column command (ESC-c, F3) and select the rectangular area of interest by moving the selection markers first vertically (unless the area is just one line deep) and then horizontally. Once the rectangle is satisfactorily defined, use the Column command's Copy subcommand (c, CTL-c) to copy the defined rectangle into the Column Buffer. From there, the rectangle can be pasted at any arbitrary location in the file. Note that rectangles can include spaces beyond the end of a line's text – VE autofills the blank areas with spaces in the copied rectangle (this is not done in the file itself).

Once copied into the Column Buffer, the rectangle can be pasted anywhere else in the file using the Column command's Paste subcommand (p, CTL-v). VE's global Paste command (CTL-v, ESC-p, F7) will accomplish the same result. For full details on the Column command, see the later section of this document “Operating on Column Areas”.

Like the line and block oriented copy commands, rectangles can also be accumulated in the Column Buffer using the Column command's Append (a, CTL-a) subcommand. There is no requirement for the rectangles to be of similar size. VE handles accumulation of differently sized rectangles with no issues.

12.3 Pasting Text Within and Between Files

VE supports pasting of lines, portions of lines (words, etc), blocks of lines (this includes paragraphs) and rectangular areas.

In all cases, previously cut or copied text may be pasted into any file currently open for editing. This means that copy/paste (CTL-c, CTL-v) or cut/paste (CTL-x, CTL-v) can be used to move lines, blocks or rectangles from one file to another. Simply cut/copy the text in one file, switch to another open file and then paste it there. This works because VE's Line, Block and Column buffers are global resources and can be used by any open file.

Pasting of each supported cut/copy type is described below.

12.3.1 Pasting Lines

Previously cut/copied individual lines can be pasted to any arbitrary location in any currently open file.

Pasting is accomplished with the Paste Line command (ESC-V) or VE's global Paste command (CTL-v, ESC-p, F7). In this case, this command will paste the contents of the Line Buffer elsewhere in the file. If the line was Cut into the Line Buffer, the Paste completes a move of the line of interest.

Multiple lines can be copy/pasted or cut/pasted in one step by preceding the CTL-c/CTL-x commands with a repetition count. Hence:

ESC 5 CTL-c

will copy the line the cursor is on, and the four lines after it, placing all five into the Line Buffer. The Paste command will then paste all 5 of them in one operation.

Copying, Cutting and Pasting of single lines can also be accomplished using the Block command (ESC-b, F2) and not moving the selection markers (this selects just the one line the cursor is on). Doing this and then using the Block command's Delete subcommand (d or CTL-x) will also cut a single line, but into VE's internal Block Buffer instead of the Line Buffer. For full details, see the later section of this document “Operating on Blocks of Text”.

12.3.2 Pasting Portions of Lines

Portions of lines can be copied, cut and pasted using the Column/Rectangle command. Using this command, portions of lines can be copied/moved from one location to another in a single file, or between open files.

A primary reason for wanting to copy/paste a portion of a line is to capture an instance of a long or difficult-to-type word (or short set of words) and then paste it everywhere else it is needed, vs. having to retype it each time.

The Column/Rectangle command is used for this. A single word, or a portion of any single line, can be thought of as a rectangular area that is simply one line deep. Hence to copy a single word, or a small set of words, using the Column command select just the portion of the line that is of interest (move directly into column selection – no need for an initial line selection in this case) and use the “c” (or CTL-c) subcommand to copy it into the Column Buffer. From there, it can be repeatedly pasted into as many places as desired using the Column command's Paste (p, CTL-v) subcommand. VE's global Paste command (ESC-p, F7, CTL-v) will accomplish the same result. For full details on the Column command, see the later section of this document “Operating on Column Areas”.

12.3.3 Pasting Blocks of Lines

Previously cut/copied blocks of lines can be pasted anywhere in any open file using the Block command.

To do this, start the Block command (ESC-b, F2) and select the continuous set of lines of interest. Then use the Block command's Copy or Delete (c or d, CTL-c or CTL-x) subcommands to copy or delete the entire block of lines into the Block Buffer. From there, the block of lines can be pasted anywhere else in the same file or another open file by using the Block command's Paste (p, CTL-v) subcommand. The same thing can be accomplished using VE's global Paste command (CTL-v, ESC-p, F7).

For full details on the Block command, see the later section of this document “Operating on Blocks of Lines”.

12.3.4 Pasting Paragraphs

VE understands the concept of paragraphs and is able to jump forward and backward a paragraph at a time (or multiple paragraphs at a time). Paragraphs can thus be copied/cut and then pasted using the Block command. This is because VE treats paragraphs as a type of block of lines, and thus paragraph copy/cut/paste is accomplished using the line-oriented Block command.

To do this, start the Block command (ESC-b, F2) and select the paragraph of interest by using the Paragraph Up or Paragraph Down commands (ESC-[, ESC-]) to move the selection markers one paragraph at a time. Then use the Block command's Copy or Delete (c or d, CTL-c or CTL-x) subcommand to copy or delete the paragraph(s) into the Block Buffer.

At this point, the paragraph(s) can be pasted anywhere else in the current file or in any other open file using the Block command's Paste (p, CTL-v) subcommand. The same thing can be accomplished using VE's global Paste command (ESC-p, F7, CTL-v).

See the “Operating on Blocks of Lines” chapter later in this document for full details on the Block command.

12.3.5 Pasting Rectangular Areas

Rectangular areas of text can be copy/cut/pasted using the Column command.

To do this, start the Column command (ESC-c, F3) and select the rectangular area of interest by moving the selection markers first vertically (unless the area is just one line deep) and then horizontally. Once the rectangle is satisfactorily defined, use the Column command's Copy or Delete subcommands (c/d, CTL-c/CTL-x) to copy or delete the defined rectangle into the Column Buffer. Note that if Delete is selected, this has the effect of copying the defined rectangular area into the Column Buffer.

Also note that rectangles can include spaces beyond the end of a line's text – VE autofills the blank areas with spaces in the copied rectangle (no change is made to the file itself).

At this point, the rectangle can be pasted anywhere else in the current file or any other open file by using the Column command's Paste subcommand (p, CTL-v). VE's global Paste command (CTL-v, ESC-p, F7) will accomplish the same result. For full details on the Column command, see the later section of this document “Operating on Column Areas”.

One thing to be aware of when pasting rectangular areas is that there must be at least as many lines in the file below the point at which the rectangle is to be pasted as there are lines in the rectangle itself in order for the paste to succeed. So for example, if a 20 line x 30 column rectangle was selected and copied and then an attempt was made to paste it at the second last line of the file, that paste attempt would fail. However, if that same paste attempt was made at the 20th last line of the file, that attempt would succeed.

12.3.6 Intelligent Paste (VE's Global Paste Command)

As has been described earlier, VE's global Paste command (CTL-v, ESC-p, F7) can be used to paste lines, portions of lines, blocks of lines, paragraphs and rectangles.

This one command can paste all of these things because VE has expanded the concept of “paste” into “intelligent paste”. The Paste command will paste the **last** object that was copied or cut, whether that object was a line, a word, a portion of a line, a paragraph, a block or a rectangle. VE remembers the last copy/cut and the Paste command always pastes the last thing that was copied or cut. This is an excellent time saver, making it unnecessary to go back into the Block or Column commands just to paste something that had previously been copied/cut there.

No matter what has been copied/cut however, there remain three unique buffers, the Line Buffer, the Block Buffer and the Column Buffer, and the contents of each one can be uniquely pasted using the associated paste commands (ESC-V for Line paste, ESC-b,p for Block Paste, ESC-c,p for Column Paste)

13.0 Moving Around In A File

13.1 Scroll Screen Up One Page, Down One Page

The Page Up and Page Down keys move the VE display page up one page or down one page respectively. If the top of file is encountered, the cursor is left at the very top of file; if the bottom of file is encountered, the cursor is left at the very bottom. These commands are also available as ESC-@ and ESC-F respectively, and again as CTL-o (previous page) and CTL-n (next page).

13.2 Scroll Screen Half Page Up, Half Page Down

The ALT+PageUp command (hold down ALT and Page Up at the same time) will move the display page up by half a page. Correspondingly, the ALT+Page Down command (hold down ALT and Page Down at the same time) will move the display page down by half a page. These command are also available as CTL-PgUp, CTL-PgDn, ESC-- and ESC-+, and again as ALT– and ALT-+.

13.3 Scroll Screen Down by an Arbitrary Amount

The VE Pick command can be used to scroll the screen downwards an arbitrary user-selected distance. VE 4.0 now calls this Vertical Pick, since its horizontal equivalent is newly supported in VE 4.0 (see below).

When the Vertical Pick command is executed, the line/row that the cursor is on is moved to the top line of the display, thus scrolling downwards in the file. This provides the user with the maximum view of what lies immediately beneath that line. For example, this might allow a user to view an entire paragraph, or the entire scope of a program's if clause, on one screen, instead of being split between screens.

To execute Vertical Pick, ensure that the cursor is at the leftmost edge of the display and enter the Pick command (ESC-z, ALT-Keypad/). The line the cursor is one will be moved to the top of the display. The command letter 'z' is patterned after vi's 'z' command, which does the same thing.

13.4 Move Selected Line to Top of Screen

Please see the above section about VE's Vertical Pick command.

13.5 Scroll Screen Half Right or Half Left

The “Scroll Right” command (ALT+Cursor Right) will scroll the display page a half screen to the right. Similarily, the “Scroll Left” command (ALT+Cursor Left) will scroll the display page a half screen to the left (if it has been previously shifted to the right, either by text entry, Cursor Right, or the Half Right command). The Scroll Left and Scroll Right commands are also available as ESC-(and ESC-) respectively.

The display screen can be scrolled completely to the left at any time by pressing the Home key, which always moves the display back to column zero of the line the cursor is on.

The scroll indicator “<-” will appear in the status line Options area when the display page is scrolled right. It disappears when the display page is scrolled back to column zero.

13.6 Scroll the Screen Right by an Arbitrary Amount

The VE Pick command can be used to scroll the screen right by an arbitrary user-selected amount as well. Historically, the VE Pick command has been used to move the current line to the top of the display page (perform vertical downward scrolling by an arbitrary amount). When the Pick command was executed, the **row** that the cursor was on was moved to the top of the display, providing the user with the maximum view of what lay immediately beneath that line, but always moving downward in the file.

In v4.0, the Pick command has been enhanced to also support horizontal scrolling, but always to the right. When Horizontal Pick is executed, the **column** the cursor is on is moved to the leftmost edge of the display, scrolling the display to the right. This allows the user to select any column (typically an indentation level) of interest and see the maximum view of what lies to the right. For example, if a line of code is heavily indented, executing Horizontal Pick on the first character of the line may allow the user to see the entire line of code, where normally it might run off the right hand edge of the screen due to its indentation level.

To execute Horizontal Pick, place the cursor on any column except for the leftmost edge of the display (because this would have no effect for Horizontal Pick) and execute the Pick command (ESC-z, ALT-Keypad/). The display window will be shifted to the right such that the column the cursor was on at the time of command execution is now moved to the left hand edge of the screen.

13.7 Move Selected Column to Left Edge of Screen

Please see the above section about VE's Horizontal Pick command.

13.8 Move Cursor Left, Right, Up and Down

The keyboard cursor keys do the expected things, moving the cursor around the file one row or column at a time. Some special behaviors are noted below.

If Cursor Left is pressed while on the first character of a line, VE will back up to the last character of the preceding line. The inverse is not true for Cursor Right at the end of a line, as VE allows you to cursor out beyond the end of a line and start typing. When this occurs, VE fills the intervening columns with spaces.

If Cursor Right is pressed when the cursor is at the right hand edge of the screen, VE will scroll the display right by one column. Similarly, if Cursor Left is pressed when the cursor is at the left hand edge of the screen, and the display has previously been scrolled right, VE will scroll the display left by one column.

The cursor left and right, up and down commands are also available via CTL-l, CTL-r, CTL-u and CTL-d, and also via ESC-, and ESC-., ESC-^ and ESC-&, respectively. Note that the cursor left and right ESC command keys are side by side on the keyboard; hence “,” is left and the key immediately to the right of it, “.”, is right. This is a commonly repeated theme in VE keyboard command mappings, whereby related command will be assigned to logically adjacent keys.

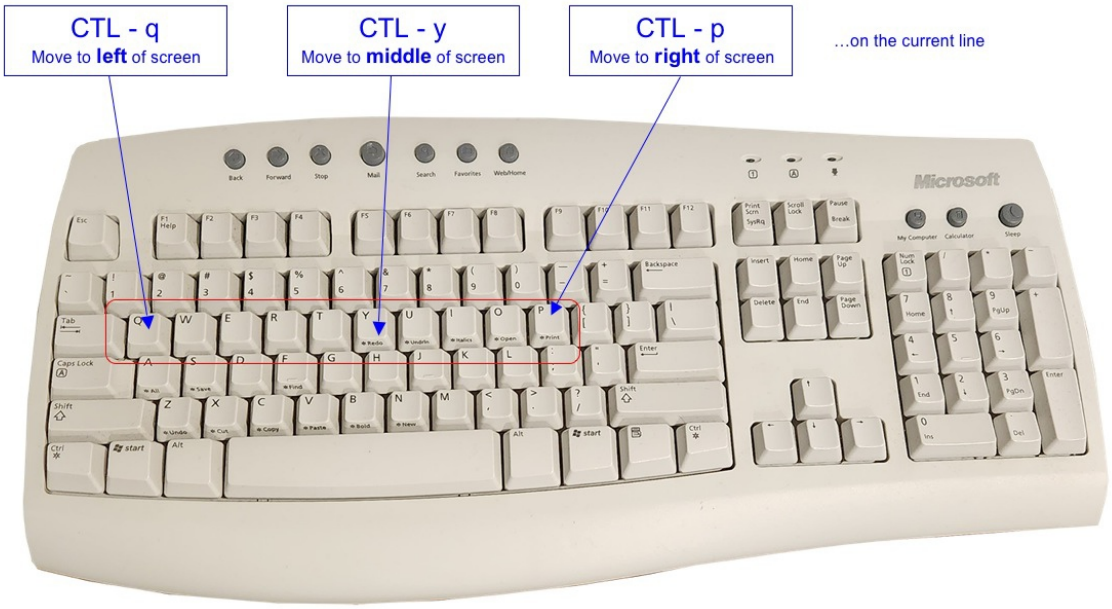
13.9 Move Cursor to Start of Line, End of Line

The HOME and END keyboard keys move the cursor the start and end of the line the cursor is on, respectively. The END key will scroll the display window to the right in order to show the end of the line, if that end lies beyond the present right hand edge of the window. Similarly, the HOME key will scroll the display window left as needed in order to display the start of the line. These commands are also available as ESC-< and ESC->.

In a nod to the venerable vi text editor, VE also supports moving to the start/end of line via ESC-0 and ESC-$ respectively, mimicking the vi mappings for these functions.

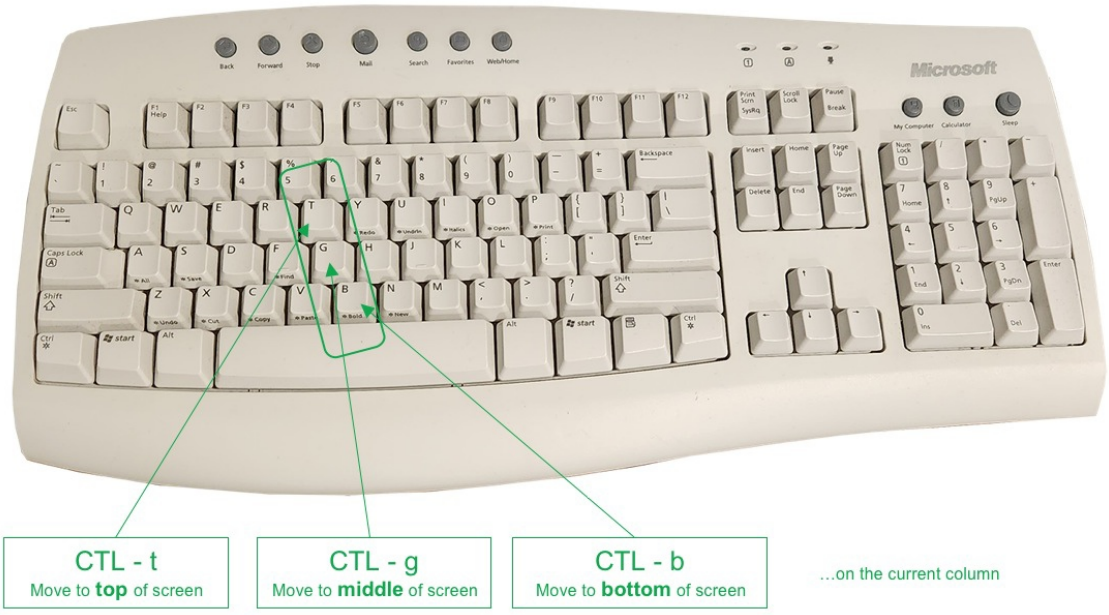
13.10 Move Cursor to Left, Middle, Right of Display

VE supports moving the cursor the left edge of the display, the middle of the display and the right edge of the display via the CTL-q, CTL-y and CTL-p commands. These initially odd-looking command mappings make more sense when viewed in the context of where these keys are on a standard QWERTY keyboard: all three are on the top row of alphabetic characters, with “q” being the leftmost alphabetic key, “p” being the rightmost alphabetic key and “y” lying roughly in the middle between these two. The command key assignments are spatially-oriented to make them easier to remember.



13.11 Move Cursor to Top, Center, Bottom of Display

VE can jump the cursor to the top of the display, the bottom of the display (or to the end of the displayed file, whichever occurs first on the currently displayed screen) and the center of the display, via the CTL-t, CTL-b and CTL-g commands. Like the commands to move to the left, right and middle of a line, these command key assignments are based on the spatial arrangement of the three keys chosen, although two of them (“t” and “b” make logical sense as well). The “t”, “g” and “b” keys on a standard QWERTY keyboard are in an almost vertical line, with “t” being at the top, “b” being at the bottom and “g” lying in the middle between “t” and “b”. Viewed this way, these key assignments are easily remembered and quickly become natural to use.



The top, middle and bottom commands are also available as ESC-H, ESC-M and ESC-L (note that these are upper case command letters). Users familiar with the vi text editor will recognize these command letters (H, M and L as being the same command letters used for the equivalent purpose in that editor.

13.12 Move Cursor Using Extended Cursor Motion

VE was originally written in the mid 1980's, before the IBM PC architecture became ubiquitous. Most keyboards of the day were not of the 102/110 key “extended PC keyboard” type. Instead, they had the basic QWERTY keys and four cursor motion arrow keys. Some had a HOME key, some did not. The computer on which VE was born, the NABU 1600, came with an ESPRIT3 serial terminal (no graphics!) with no Page Up or Page Down keys, but four cursor movement arrow keys clustered like the four points of the compass around a central HOME key. This is nearly identical to the configuration of the keypad on most standard PC keyboards today, with the four cursor motion keys being located on the 2, 4, 6 and 8 keys, with the 5 key in the middle.

That key configuration gave rise to the Extended Cursor Motion command. When a cursor key is used, it sets a cursor direction. The Cursor Left key sets the direction to left, the Cursor Right key sets it to right and so on. When the HOME key was pressed, the cursor performed an “extended motion” in the direction just set. In the Qunix original, these four directions were used to achieve the equivalents of Page Up, Page Down, Home and End.

The original VE Extended Cursor Motion command is still available in VE today, bound to the CTL-e key. When a cursor key is used, it sets a cursor direction. The Cursor Left key sets the direction to left, the Cursor Right key sets it to right and so on. When CTL-e is pressed, VE performs an extended cursor movement in the currently set cursor motion direction:

- Cursor Left and then CTL-e moves the cursor to column 0 of the line it is on, the logical equivalent of today's Home key.
- Cursor Right and then CTL-e moves the cursor to the end of the line it is on, the logical equivalent of today's End key.
- Cursor Up and then CTL-e moves up one screen in the file, the equivalent of today's Page Up key.
- Cursor Down and then CTL-e move down one screen in the file, the equivalent of today's Page Down key.

The Extended Cursor Motion command acquires its real utility in that it remembers the current cursor motion direction, and repeated use of the CTL-e key will continue to move in the indicated direction. Hence, Cursor Up, followed by repeated CTL-e keys will page up through the file, equivalent to repeatedly selecting the Page Up key.

This can be made even more useful with the Again command. After selecting cursor up or down and then CTL-e, the Again command (conveniently mapped to the F1 key, and also ESC-a) can be used to repeat the movement in the same direction. Hence, the Extended Cursor Motion command still supports convenient paging up or down through a file.

The Extended Cursor Motion command is little used today, but is retained in VE in deference to “the old days” (somewhat like the Power Typing option described earlier).

13.13 Move Cursor To Display Edge

The current version of VE uses the same concept of setting and acting on directions today to move to a screen edge, since today's PC keyboards provide dedicated keys for the Page Up, Page Down, Home and End functions. When the Edge command is used (bound to keypad “5” key), the cursor moves to the screen edge indicated by the currently set cursor motion direction.

This works as follows:

- Cursor Left and then “Keypad 5” moves the cursor to the left edge of the display. This may not be the first column of the file if the display page has been scrolled right.
- Cursor Right and then “Keypad 5” moves the cursor to the right edge of the display.
- Cursor Up and then “Keypad 5” moves the cursor to the top line of the display (the same effect is available via the Move Cursor to Top of Display described earlier).
- Cursor Down and then “Keypad 5” moves the cursor to the bottom line of the display page (the same effect is available via the Move Cursor to Bottom of Display command described earlier).

13.14 Move Cursor to Next Word, Previous Word

The ESC-CursorRight (ESC and then the cursor right key) and ESC-CursorLeft (ESC and then the cursor left key) commands will move forward or backward by one word. These commands are also available via ESC-}, ESC-{ respectively.

Note that to move over several words sequentially, it can be tedious to continually type ESC-CursorRight. Instead, after the first one, move over successive words by typing F1 (Again command).

13.15 Move Cursor to Next Para, Previous Para

The ESC-CursorDown (ESC and then the cursor down key) and ESC-CursorUp (ESC and then the cursor up key) commands move the cursor to the bottom of the current paragraph or the top of the current paragraph. Repeated use of these commands will move up and down the file a paragraph at a time. These commands are also available via ESC-] and ESC-[respectively.

Note that to move over several paragraphs sequentially, it can be tedious to continually type ESC-CursorDown. Instead, after the first one, move over successive paragraphs by typing F1 (Again command).

13.16 Move to Top of File

The ESC-PageUp command moves the display window to the top of a file. This command is also available as ALT-Home and ESC-/. The same effect can also be achieved via the [Jump, Start] menu path (ESC-j, s).

13.17 Move to Bottom of File

The ESC-PageDown command moves the display window to the bottom of a file. This command is also available as ALT-End and ESC-Z (note the “Z” is capitalized). The same effect can also be achieved via the [Jump, End] menu path (ESC-j, e).

13.18 Jump Start, End, Line Number

General:

The ESC-j command allows jumping to the start, to the end, or to a specified line number of a file. This command presents a menu containing “Start”, “End”, “Line” and “Tag” selections. “Start” and “End” will move to the start and end of the file, respectively. The “Line” selection will prompt for a line number and then jump to that line. Line numbers in VE start at Line 1. The “Tag” selection is documented in the Tags section below.

Shorthand Jump to Line:

As a convenience, VE also supports a slightly faster way of jumping to a line number, per:

ESC line_number j

Normally, “ESC n command” is a request to run the given command “n” times. However, since it does not make sense to repeatedly jump to the same line, this key combination is repurposed for the Jump command to accomplish in slightly fewer keystrokes the function of jumping to the line number specified in the repetition count.

Note that lines numbers in VE start at 1, so “ESC 0 j” will NOT jump to the start of the file. The command for that would be “ESC 1 j” (jump to line 1) or “ESC j s” (jump to start of file). In fact, “ESC 0” is bound to the “start of line” command in a nod to a well-known vi key binding.

Jump to Line at Startup:

Note the VE also supports jumping directly to a line number upon startup, via the startup command form:

ve/ve88/ve32 filename -l line_number

13.19 Tags, and Jumping to Tags

VE supports a simple mechanism for tagging lines, and then jumping to a defined tag at a later time. This can be useful when it becomes necessary to check something in another part of a file being edited. In this case, the current line can be tagged, so that it may be returned to after other parts of the file have been viewed. The tagged line can be returned to at any time by simply jumping to the previously defined tag.

Defining Tags:

VE supports 10 named tags, named “0” to “9”. To tag the line the cursor is presently on, simply enter ESC-t and then a tag number, from “0” to “9”. VE will respond with a message indicating that the named tag has been set. Setting a tag that has already been set silently redefines the tag – VE does not complain about overwriting existing tags. This is intended operation.

Listing Tags:

The set of defined tags for the current file can be viewed by executing the [Tags, List] command (ESC-t, l menu path). VE will list all 10 tags, showing the associated line number, or “Undefined” if they have not been set.

Jumping to Tags:

VE supports two mechanisms for jumping to tags. The first and most obvious of these is the [Tags, Jump] command (ESC-t, j) which prompts for a tag number (“0” to “9”) and then jumps to that tag. The second and perhaps equally obvious mechanism is via the [Jump, Tag] command (ESC-j, t).

Tags and Multiple Files:

Tags are associated with a specific file. Each file that VE has open has its own unique tag definitions for each of the 10 available tags. Unlike undo items, you may freely move between files without worry about losing tag definitions.

Limitations of the Tags Function:

There are two salient limitations of VE's simple Tags functionality:

- Tags are dynamic associations that only last as long as VE has a given file open. They are not stored with the file. Hence, when a file is closed, all tags for that file are discarded.
- Tags are are a simple means of associating a line number with a tag symbol. The association is with the line number, not with the line itself. If a tag is defined and then the number of lines in the file is changed via text deletion, insertion, pasting, etc. the actual line that a given line number points to will be changed. Hence a jump to a tag may not achieve a jump to the expected line.

13.20 Current Line, Column Information

The Info command, available via the ESC-i or ALT+i keystrokes, displays VE's current position in the file, including line, column and percentage through the file. It also displays the total number of lines in the file.

A number of other VE commands, such as Find, Page Up and Page Down, also present the same information, dynamically updating it as they move through the file.

14.0 Finding and Replacing Text

VE provides a Find command for finding text and a Replace/Substitute command for finding and then replacing text.

14.1 Finding Text

The VE find command (ESC-f, F5) finds an occurrence of a specified string and moves the cursor to the start of that string. When ESC-f or F5 is entered, VE provides a “Find:” prompt for entry of the string to find. This prompt will be preloaded with the last find string specified if a Find command has been executed at an earlier time in the editing session. The existing preloaded string may be directly re-used by simply pressing ENTER, or may be edited as desired (see the section earlier in this document on Status Line Response Line Editing for the full set of line editing features supported), or may be replaced with a new string (enter F10 or ESC-k to clear the existing string and then enter a new string). VE will search the file for the next occurrence of the specified string and move the cursor to the start of that occurrence.

When an instance of the find target is located, VE moves the screen display to that target and updates the Status line with information about the line number of the target and the percentage of the way through the file it is located at.

To locate the next and succeeding instances of the find target, the ESC-f/F5 command can be retyped, or the VE Again command (F1, ESC-a) can be used to repeatedly re-execute the last Find command.

The VE Find command supports two special search characters, allowing searches to include the beginning of a line, or strings that occur only at the beginning of a line, and a similar functionality for the end of a line, or strings that terminate at the end of a line. This is especially useful in conjunction with the Replace command (see below).

The default start-of-line character is '^' and the default end-of-line character is '$'. These defaults may be changed via the [Options, Match] menu path (ESC-o. m), using it's Start-of-line and End-of-line subcommands.

14.2 Replacing Text

The VE Replace/Substitute command (ESC-r, F6) replaces one occurrence of a specified string with another specified string. If used with a repetition count, it will make multiple replacements sequentially. If used with the '*' repetition count, it will replace all occurrences in the file.

When this command is entered, VE will provide a “Replace:” prompt. This prompt will be preloaded with the last Replace string entered if the Replace/Substitute command has been used earlier in the editing session. Like the Find command, the existing string may be re-used directly by simply pressing ENTER, or may be edited as desired (see the section earlier in this document on Status Line Response Line Editing for the full set of line editing features supported), or may be replaced with a new string (enter F10 or ESC-k to clear the existing string and then enter a new string). VE will then provide a “with:” prompt, prompting for the string to use to replace the “Replace:” string with. Again, this will be preloaded with the last replacement value specified if the Replace/Substitute command has been used earlier in the editing session. This value may be used directly, edited to suit, or replaced, per the above. VE will then attempt the requested replacement and provide the results.

To request that VE prompt for permission before making each replacement, the letters “ v” (a space and then lower case “v”) may be appended to the “with:” string. When this is done, VE will prompt for verification of each replacement before it is made.

To replace the next and succeeding instances of the replace target, the ESC-r/F6 command can be retyped, or the VE Again command (F1, ESC-a) can be used to repeatedly re-execute the last Replace command.

Like the Find command, the Replace command supports the Start-of-line and End-of-line characters "^" and "$" in its initial Replace: prompt. These characters can be very useful by themselves as Replace targets. For example, as a replace target, "^" allows you to insert text at the beginning of each line. Similarly, "$" allows you to add text at the end of each line.

As a practical example of this, if you wish to temporarily comment out a number of C++ lines, you can issue the command:

ESC 8 r ^ //

This command runs the "r" command (the Replace command) 8 successive times, replacing the start of line for each of 8 lines with the C++ comment characters "//".

14.3 Repeating Find/Replace Operations

As mentioned above, the last find or replace operation may be repeated by simply pressing the F1 key (Again command). This can be done as many times as desired, or until no more occurrences of the search string can be found (VE will indicate this in the Status Line when it occurs).

15.0 Operating on Blocks of Lines

VE provides a complete set of operations for manipulating blocks of text lines. Two essential mechanisms are supported, line-oriented and block-oriented. Similar functions are also available for column areas (rectangular areas of text), and are documented in the next chapter.

15.1 Line Oriented Copy, Cut and Paste

VE maintains a global internal Line Buffer that is used by VE's line-oriented cut/copy/paste commands.

These commands are intended for operations on single lines or on small numbers of lines.

The following line-oriented operations are available:

- Copy Line(s) (CTL-c, ESC-C) – this command copies one or more lines to the Line Buffer. To copy just the line the cursor is on, type CTL-c. To copy up to "n" lines starting with and including the line the cursor is on, type "ESC n CTL-c".

  In a nod to the venerable vi text editor, Copy Line is also bound to ESC-y (Yank).

- Append Line(s) (CTL-a) – this command is identical in operation to the Copy Line (CTL-c) command above except that instead of replacing any prior contents of the Line Buffer with the line the cursor is on, it adds the line the cursor is on to any existing lines already in the Line Buffer.

  This allows users to accumulate individually selected lines (or small groups of lines) into the Line Buffer prior to pasting them somewhere else.

- Cut Line(s) (CTL-x, ESC-X) – this command deletes one or more lines after copying them to the Line Buffer. To delete just the line the cursor is on, type CTL-x. To delete up to "n" lines starting with and including the line the cursor is on, type "ESC n CTL-x".

- Paste Line(s) (CTL-v, ESC-V) – this command pastes the contents of the Line Buffer into the file, above the line the cursor is presently on, except if the cursor is on the last line of the file. In this case, the contents of the Line Buffer are pasted below the current line.

  In another nod to the venerable vi text editor, Paste Line is also bound to ESC-p (put). Hence vi's yank/put sequence for copy/paste lives on in VE as well.

Per the above, any one of these commands can be preceded with a repetition count to operate on multiple lines at once. As an example:

ESC 5 CTL-c

will copy 5 lines into the Line Buffer.

The remainder of this chapter deals with VE operations oriented towards blocks of text lines (referred to as a "line block" or simply a "block").

15.2 Defining a Line Block

VE maintains a global Block Buffer that is used by VE's block-oriented cut/copy/paste commands.

The VE Block command (ESC-b, F2) brings up the Block menu, which presents a menu of possible block operations and a visual marker on screen at the start of the line the cursor is presently on. Note that line blocks are complete lines – the cursor is moved to the start of the line to emphasize this. To perform the equivalent copy, cut and paste operations on partial lines use the Column Area commands (documented in the next chapter).

The visual marker is a (typically red colored) right carat symbol, ">". To define the block of interest, use cursor motion commands to move the marker. As cursor motion commands are used to move away from the first line of the block, a second marker will appear. All lines between the two markers (including the lines that the markers are on) define the current block. The "cursor up/down", "page up/down", "half page up/down", "paragraph up/down", "jump to line", "jump to tag" and "jump to start/end of file" commands may be used in any combination to position the second marker, thus defining a line block.

As a convenience, while defining a block the paragraph up/down and the "jump" commands do not have to be preceded with ESC. The Block command understands "]" as the paragraph down command, "[" as the paragraph up command and "j" as the Jump command.

At any time during the definition of a line block, it is possible to "escape" out of the Block command entirely by simply pressing ENTER.

15.3 Line Block Copy, Cut, Paste, Wrap, Read, Write

Block-oriented copy, kill, cut and paste operations are intended for convenient manipulation of larger blocks of text lines. A block of text lines is defined as one or more full lines. VE operations are provided to visually define a line block (see above), copy it to the Block Buffer, append it to the Block Buffer, kill it, delete it (which also copies it to the Block Buffer), paste the contents of the Block Buffer, and write the Block Buffer out to a file. An operation is also provided to read in an external file into the Block Buffer, and then paste those contents into the file. VE also provides a block operation to perform after-the-fact text wrapping of a whole line block at once. Finally, VE provides an operation to exchange the line-oriented Block Buffer and the rectangle-oriented Column Buffer, allowing line block operations to be performed on rectangle/column blocks and visa versa.

The following block-oriented operations are available. Type the first letter of the command to execute it:

- Copy Block ("c" subcommand of Block Menu, also CTL-c) – this command copies the selected block into the Block Buffer. This command can be used in conjunction with the Paste Block command (below) to copy a block of text from one part of a file to another. Simply copy the block using the Copy Block command, then move to the intended target area and use the Paste Block command (below) to paste the block into the file.
- Append Block ("a" subcommand of Block Menu) – this command is similar to the Copy Block command (above), but has the added attribute that the defined line block is **appended** to the current contents of the Block Buffer, as opposed to replacing any current contents, which is the behavior of the Copy Block command. The Append Block command therefore allows users to **accumulate** sections of text into the Block Buffer before pasting them elsewhere or writing them out to a file. This is similar in intent for blocks to the CTL-a command for lines.
- Kill Block ("k" subcommand of Block Menu) – this command deletes the selected line block from the file **without** saving a copy of it in the Block Buffer.
- Delete Block ("d" subcommand of Block Menu, also CTL-x) – this command deletes the selected line block from the file after first copying it to the Block Buffer. This command can be used in conjunction with the Paste Block command (below) to move a block of lines from one part of a file to another. This can be done by deleting the line block using the Delete Block command, then moving to the intended target area, and then using the Paste Block command (below) to paste a copy into the file at the new location.

- Paste Block ("p" subcommand of Block Menu, also CTL-v) – this command pastes the contents of the Block Buffer into the file, above the line the cursor is presently on. An exception to this occurs if the cursor is on the last line of the file. In this case, VE will paste the Block Buffer contents into the file below the line the cursor is on.
- Wrap Block ("w" subcommand of Block Menu) - this command performs after-the-fact text wrapping on the defined line block. The entire block will be wrapped to fit between the defined left and right margins. Note that this command is paragraph-oriented. If the defined line block does not enclose complete paragraphs, this command will find the paragraph start nearest to the start of the block, and the paragraph end nearest to the end of the block, and wrap that complete range.

  This command operates on the defined line block in the file and does not impact any then current contents of the Block Buffer.
- Read Block ("r" subcommand of Block Menu) – this command presents the VE Open File Dialog to allow selection or entry of a filename, reads that file into the Block Buffer, and then pastes the result into the currently editing file above the line the cursor is presently on. In essence, this command provides a way of **inserting an external file** into the file currently being edited. Like the Paste command, an exception occurs if the cursor is on the last line of the file – in this case the paste will occur below the line the cursor is on.
- Write Block (Save As) ("s" subcommand of the Block Menu) – this command writes the currently defined block **or** the contents of the Block Buffer out to a separate file. This command presents the VE Open File Dialog to prompt for a filename, and then writes the currently defined line block or the contents of the block buffer out to the indicated file.

  The filename may be directly typed, or users may select an existing file to overwrite. Partial filenames can be entered and completed using auto-completion. VE will prompt before overwriting any existing file. The block may be written to directories other than the current directory by either typing the full or partial (with auto-completion) pathname to the new directory as part of the filename, or using the Open File Dialog to navigate to the directory of interest before typing a filename.

  This command incorporates a useful subtlety, allowing a block of text lines to be written out to a file without disturbing the current contents of the Block Buffer.

  If a block is defined using the Block Command (ESC-b, F2), and then the Write Block subcommand is immediately used next, the currently defined block will be written out to the requested file, but the present contents of the Block Buffer will not be effected.

  If on the other hand the block is defined using the Block Command (ESC-b, F2) and then copied to the Block Buffer, a later [Block Write] will write out the contents of the Block Buffer as expected.

  The "useful subtlety" is that areas of your file can be written out to separate files without impacting the contents of the Block Buffer if so desired.
- Flush Block ("f" subcommand of the Block Menu) – this command does as its name suggests, emptying (flushing) the contents of the Block Buffer. This may be useful when editing a large number of large files, or when a lot of text lines have been placed into the Block Buffer. This command frees the associated memory, thus assisting in low memory situations.
- Exchng Block ("e" subcommand of the Block Menu) – this command exchanges the contents of the line-oriented Block Buffer with the contents of the rectangle/column oriented Column Buffer, so that rectangle/column operations can be performed on line buffers and visa versa.

15.4 Repeating Line Block Commands

Like most VE commands, the Block command (ESC-b, F2) can be used with repetition counts to achieve multiple executions with a single user command. Alternately, the VE Again command (ESC-a, F1) can be used to repeat the last executed Block command as many times as desired. When used with a repetition count, only the initial repetition will interact with the user to define the line block. All subsequent executions will use the block defined in the first repetition. The same is true of repetitions accomplished via the Again command. Executing Again (ESC-a, F1) after a Block command will re-use the last Line Block definition without requiring the user to define the same block again for each repetition.

15.5 VE Block Paste Command Revisited

As an added convenience, VE makes the Block Paste command available via the global Paste command (CTL-v, ESC-p, F7). This command runs VE's Intelligent Paste command, which will paste the last object that was copied. If that was a line block, via any of the above, Intelligent Paste runs the Block Paste command (without requiring the user to enter the Block command (ESC-b, F2) first). This minimizes the number of keystrokes needed for this very common operation (pasting copied lines). The VE Paste command, like the [Block, Paste] command, can be used with repetition counts, or with the Again command. For example:

ESC 6 p

will paste 6 successive copies of the contents of the internal block buffer into the file.

16.0 Operating On Column Areas (Rectangles)

In addition to support for manipulation of blocks of text lines, VE also provides the powerful capability to manipulate areas of text columns and rows as a single entity. This entity is referred to throughout this document as a “column area”, or a “rectangle”. VE maintains a global Column Buffer that is used by VE's column/rectangle-oriented cut/copy/paste commands.

A “column area” or “rectangle” is a rectangular area of text consisting of one or more columns spanning one or more lines. A column area can be as small as a single character on a single line, or as large as all of the characters on all of the lines of the file. Using column areas, VE can cut and paste single words, partial lines of text, whole columns of data (for example in column oriented output of a data reporting tool) or whole files. The ability to operate on column areas / rectangles is an extremely flexible and powerful capability within VE.

Column areas / Rectangles always act on the existing lines of a given file – they can alter existing lines, but they cannot be used to create new lines. In essence, VE's rectangle operations allow users to insert, copy, cut and paste horizontally, while VE's line block operations allow users to insert, copy, cut and paste vertically. To bridge the gap between these, VE also provide the ability to exchange the line block buffer and the column area buffer, allowing column area operations to be performed on line blocks, and visa versa.

16.1 Defining a Column Area (Rectangle)

After entering either ESC-c or F3, VE will execute the Column command, which will present the Column menu and place a visual marker onto the screen at the current cursor position. Defining a column area is a two step process. First, the horizontal range (a line range) is defined, and then the vertical range (a column range) is defined. These two in combination specify a rectangular area within the file that will be the target for column/rectangle operations.

The definition of a column area is similar in operation to the definition of a line block. Column Area definition is carried out interactively by moving visual markers on the display page to define first the line range and then the column range.

Define the Vertical (Line) Range

For defining the line range, the visual marker is the same (typically red colored) right carat symbol (“>”) used in line block definition. To define the line range of interest, use cursor motion commands to move the marker. As cursor motion commands are used to move away from the first line of the range, a second marker will appear. All lines between the two markers (including the lines that the markers are on) define the line range of the column area. The “cursor up/down”, “page up/down”, “half page up/down”, “jump to line”, “jump to tag” and “jump to start/end of file” commands may be used in any combination to position the second marker, thus defining the line range for the column area.

Define the Horizontal (Column) Range

When the line range has been satisfactorily defined, use of any horizontal cursor motion key (for example cursor right or cursor left) will begin column range definition. VE will return the cursor to the initial position it was at when the Column command was first executed (the upper left or right corner of the rectangle being defined), and replace the displayed line markers with column markers. If the defined line range is only a single line, only one column marker will be displayed. The column marker is a (typically red colored) “down carat” symbol pointing to the column of interest. In some cases, VE may use the letter “capital v” - “V” instead.

To define the column range of interest, horizontal cursor motion commands are now used to move the column markers. As cursor motion commands are used to move away from the first column of the range, a second set of markers will appear. All columns between the two markers on each line (including the columns that the markers are on) define the column range of the column area. All text enclosed within the four displayed markers is included in the defined column area. Note that if the defined line range for the column area is only a single line, just two markers will appear. The “cursor left/right”, “Home/END”, Left Edge, Right Edge, “Middle of Screen” and finally the “Word Left/Right” commands may be used in any combination to position the second set of column markers, thus defining the column range.

Column/Rectangle Definition Shortcuts

The method for defining a column area may be abbreviated where this makes sense. For example, if the area of interest is a single word or phrase on the current line, it is possible to go directly to column range definition without defining a line range. To do this, simply enter a cursor left/right key as the first keystroke after the Column command presents the line range definition marker. VE will immediately move to column definition, changing the line marker to a column marker. Similarly, if the area of interest is only one column wide, simply enter the first letter of any of the Column command's menu items immediately after defining the line range and pressing cursor left/right to enter column range definition.

In its simplest form, the definition of a column area can consist of one cursor left/right key, followed by a Column command. This defines and operates on a single character column area!

Note that at any time during the definition of a column area, it is possible to "escape" out of the Column command entirely by pressing ENTER.

Select a Column/Rectangle Command

When the column range has been satisfactorily defined, a full rectangle has been specified. To terminate column area definition and execute a command on the defined column block, enter the first letter of any selection from the Column command's menu (which has been continuously presented on the status line during the column block definition process). Once a command has been entered, VE will remove all displayed markers and execute the entered command.

16.2 Column Area Insert, Copy, Cut, Paste, Move, Read, Write

VE provides a full set of Column commands to operate upon a defined column area / rectangle. Commands are provided to insert a blank column area (useful for indenting lines of text), kill a column area (conceptually the inverse of the above), copy a column area to the Column Buffer, delete a column area (also copies it to the Column Buffer), paste a column area from the Column Buffer, read a column area in from an external file into the Column Buffer and then paste it, write the defined column area out to an external file, empty the Column Buffer and finally, exchange the Column and Block (line) buffers. Each of these capabilities is detailed below.

- Insert Columns ("i" subcommand of Column menu, also the keyboard Insert key) – this command inserts a number of blank columns into the file starting at the cursor position at time of Column command execution. The number of columns inserted is determined by the width of the defined column range. This command is extremely useful for indenting blocks of source code, etc.

  This command may be entered without defining a column range, which results in a single column being inserted. This command can also be used without even the initial line range definition, resulting in a single space being inserted at the current cursor position.

  To insert an arbitrary number of blank columns (an arbitrary indent level), simply press the F1 (Again) command after inserting the first blank column. Continue to press F1, visually observing the effect of the increasing indentation, until the desired level of indentation is reached.

  To make this command even more convenient to use, VE supports the use of the keyboard Insert key as the Column menu subcommand for Insert Columns. Instead of pressing "i" and then F1 repeatedly, simply press Insert as many times as desired, visually observing the effect of the increasing indentation until the desired level of indentation is reached.

- Kill Columns ("k" subcommand of Column menu) – this command is the philosophical inverse of the Insert command. It deletes the defined column area without saving a copy of it to the Column Buffer. Like Insert, this command is also extremely useful for changing the indentation of blocks of source code, etc.

  Like Column Insert, this command may be entered without defining a column range, which results in a single column being deleted. Like the Insert command, this command can also be used without even the initial line range definition, resulting in a single character being deleted at the current cursor position.

Like Column Insert, this command can also be repeated with the F1 (Again) command. Like Column Insert, this command also has a convenient keyboard shortcut – the keyboard Delete key can be used as the Column subcommand for Kill Column, allowing users to simply press the Delete key repeatedly until the number of desired columns have been deleted, all the while observing the feedback in real time on the screen.

- Delete Columns (“d subcommand of Column menu, keyboard Delete Key, CTL-x) – this command deletes the defined column area in the same manner as the Kill command, but saves a copy of it in the Column Buffer prior to doing so. Any previous contents in the Column Buffer are discarded and replaced with the new column area being copied. Coupled with the Paste Columns command (see below) this is command can be used to move column areas of text to different locations in the file.

  This command may be entered without defining a column range, which results in a single column being deleted. Similar to the Kill subcommand, this command can be used without even the initial line range definition, resulting in a single character being deleted at the current cursor position. Also similar to the Kill subcommand, this command can be repeated with the F1 (Again) key, allowing users to visually observe the results of their deletion until they have achieved the desired outcome.

- Copy Columns (“c” subcommand of Column menu, CTL-c) – this command creates a copy of the defined column area in the Column Buffer. Any previous contents in the Column Buffer are discarded and replaced with the new column area being copied.

  This command may be entered without defining a column range, which results in a single column being copied. Like most other Column commands, this command can be used without even the initial line range definition, resulting in a single character being copied at the current cursor position.

- Append Columns (“a” subcommand of the Column menu) – this command operates in an identical manner to Copy Columns except that instead of replacing the current contents of the Column Buffer with the new column area being operated on, this command adds the new Column area to the bottom of any existing rectangles in the Column Buffer. Just like the Append operations for each of the Line and Block buffers, this allows users to **accumulate** content in the Column Buffer prior to pasting it in another location.

- Paste Columns (“p” subcommand of Column menu, CTL-v) – this command pastes the current contents of the Column Buffer into the file, starting at the current cursor position. The contents are pasted to the right of the current cursor position.

  This command is typically entered without defining a line and or column range. In this case, VE pastes the current contents of the Column Buffer into the file with the first column of the first line of that buffer placed at the current cursor position. If the Column Buffer contents will not fit at the current cursor position (for example, there are not enough lines between the current cursor position and the end of file) VE will indicate this condition and will not do the paste.

  Use of the Paste Columns command without an initial line and column range definition is in fact the intended and most convenient way to use this command. To copy a column area, the Copy Columns command is used, which requires a full column area definition. Then the cursor is moved to the intended target area, the Column command is run again. The “p' (Paste Columns) subcommand is then selected from the Column menu, which pastes the previously copied column area at the new location. Note that VE's global Paste command (CTL-v, F7, ESC-p) will achieve the same result.

- Read Column Area From File (“r” subcommand of Column menu) – this command presents the VE Open File Dialog to prompt for an external file name and then reads the contents of the selected file into the Column Buffer. Any previous contents of that buffer are discarded and replaced with the new content. The contents of the Column Buffer are then pasted into the file at the current cursor position.

  Conceptually and in implementation, this is simply a file read, followed by a Paste Columns command. After the filename prompt, its behavior is identical to the Paste Columns command. Like the line-oriented Block Read command, this command allows an external file to inserted into the currently editing file. The key difference between Block Read and Column Read is that with Column Read, the external file can be inserted at a column different than the left hand edge of the file (column 0).

- SaveAs – Write Defined Column Area to File (“s” subcommand of Column menu) – this command writes the defined column area out to an external file, prompting for the file's name via the VE Open File Dialog. The file's name may be fully or partially typed (with auto-completion for partially typed filenames), or an existing file may be selected to be overwritten. VE will prompt before overwriting any existing file. The defined Column Area may be written to a directory other than the current directory by either typing in the full or partial pathname of the intended directory as part of the filename, or using the Open File Dialog to navigate to the directory of interest before entering the filename. Like most other column commands, this command can be used without either the column range definition or even the line range definition, resulting in a single column or a single character being written out to file, respectively.

  Like its SaveAs counterpart in the line-oriented Block commands, this command has a subtle twist in operation which can be exploited as desired. If used with a column area definition, this command will write the defined area out to file. If used without a column area definition, this command will write the current contents of the Column Buffer out to file. Hence, it is possible to save defined column areas to file without impacting the current contents of the Column Buffer.

- eXchange Column/Line Buffers (“x” subcommand of Column Menu) – this command exchanges the contents of the Column Buffer and the Block Buffer. This allows saved column areas to be treated as full lines, and full lines to be treated as column areas, as desired. As an example of potential usage, column areas can be saved or deleted from one area of a file and pasted into another as full lines, achieving the ability to move text both horizontally and vertically. Many creative uses of the eXchange command are possible.

- Empty Column Area Buffer (“e” subcommand of Column Menu) – this command is the logical equivalent of the Flush command in the line-oriented Block menu. It empties the current contents of the Column Buffer and frees all associated memory.

16.3 Repeating Column Area Commands

Like most VE commands, the Column command (ESC-c or F3) can be used with repetition counts to achieve multiple executions with a single user command. Alternately, the VE Again command (ESC-a or F1) can be used to repeat the last executed Column command as many times as desired. When used with a repetition count, only the initial repetition will interact with the user to define the column area. All subsequent executions will use the column area defined in the first repetition. The same is true of repetitions accomplished via the VE Again command. Executing Again (ESC-a or F1) after a Column command will re-use the last Column Area definition without requiring the user to define the same column area again for each repetition.

17.0 Inserting From, Writing to External Files

17.1 Writing Selected Areas to an External File

VE can write a selected area of the file currently being edited out to an external file. Please see the [Block, SaveAs] and [Column, SaveAs] commands in the chapters “Working with Line Blocks” and “Working with Column Areas” respectively.

[Block, SaveAs] writes a selected continuous set of full lines from the file currently being edited out to an external file.

[Column, SaveAs] writes a selected rectangular area of the file currently being edited out to external file.

17.2 Inserting an External File into File Being Edited

VE can read in an external file and insert it into the file currently being edited. Please see the [Block, Read] and [Column, Read] commands in the chapters “Working with Line Blocks” and “Working with Column Areas” respectively.

[Block, Read] reads in the contents of an external file and inserts them into the file currently being edited as a series of lines.

[Column, Read] reads in the contents of an external file and inserts them into the file currently being edited as a column area. That column area can be inserted at any arbitrary column. This is different than [Block, Read], which always inserts as full lines, starting at column 0.

In the (slightly) abstract, [Block, Read] inserts the contents of external files vertically (creates new lines for them) while [Column, Read] inserts the contents horizontally (creates new columns for them).

18.0 Exchanging Lines and Rectangles

The ability to read and write both line-oriented areas and rectangle-oriented areas to/from external files can be used to copy a rectangular area and paste it as a set of lines, allowing users to extract content at any arbitrary location in a file (particularly content that is not at column 0) and paste it as a set of lines. It can also be used to do the inverse; copy a set of column 0 aligned lines and paste them as a rectangle at any arbitrary column number.

To do this, simply save the source, be that lines or a rectangle, to an external file. Then read that file back in as the inverse type. If written as lines, read in as a rectangle. If written as a rectangle, read in as lines. This simple approach allows users to insert content anywhere desired in text-based documents.

This same result can be achieved even more simply by using the eXchange subcommand of the Block and Column commands. The eXchange subcommand allows users to exchange the contents of the Block and Column buffers, effectively accomplishing the same result without the use of an external file. Copy a block of lines, exchange the two buffers, and then paste as a rectangular area. Alternately, copy a rectangle, exchange the two buffers, and then paste as a series of lines.

19.0 Undoing Changes

VE supports universal undo. The Undo command (ESC-u, ALT-u or CTL-z) provides a comprehensive Undo capability, allowing users to undo any number of changes to a file, in the order in which they were made. In the current release of VE, the undo queue is arbitrarily limited to a depth of 128 items. After this limit is reached, as new undo actions are added, the oldest ones are discarded. Users are not able to change this limit.

Like all other VE commands, the Undo command may be used with a preceding repetition count. For example, “ESC 5 u” will undo the last five changes made. “ESC * u” will undo all changes presently queued for undo action, potentially returning the file to the state it was at when opened.

The Undo command fully supports the undoing of macro executions. VE supports macros, which are interactively defined collections of VE commands that accomplish some useful purpose (the creation and use of VE macros is fully documented later in this user manual). After executing a macro, a single Undo command (ESC-u) will undo all changes that have been made by that macro, irrespective of the total number of changes it made.

The state of the VE undo system can be viewed directly at any time via VE's Status command (ESC-s). Among the information shown by this command is the number of Undo actions that are queued and the number of Undo items they are queued in.

VE's undo capability is specific to the file being edited at the time changes are made. If a user switches from one open file to another (either by opening an additional file, or by switching to another already open file), VE will flush the current queue of pending undo actions in order to preserve file continuity.

VE does not support a Redo capability. Once a change has been undone, it cannot be redone, except by manually entering the change again.

20.0 Saving Your Work

VE provides three commands for saving your work. The first is the Update command; the second and third are subcommands of the Quit Command. These allow you to periodically save your current file to disk, and then optionally return to your editing session, write the current file out to another filename or quit VE.

20.1 Saving Your File As You Work

The Update command (CTL-s, F11, ESC-u or ESC-q u) writes the contents of the file currently being edited out to disk and then returns to editing that file. This provides a convenient way to checkpoint your work as you go. This is equivalent to the [Edit, Save] dialog on most GUI editors.

For maximum user convenience, the Update command is also bound to the CTL-s keystroke, which is commonly used by many GUI programs for the same purpose.

The Update command can be used even if the current file has not yet been named (is a new unnamed file). An attempt to Update such a file will cause VE to prompt for a filename. The file will then be written out under that name, which will then become the file's current name.

20.2 Saving Your Work To Another File

The Write subcommand of the Quit command (ESC-q/F12 w) writes the current file out to another filename. This subcommand presents the VE Open File Dialog to prompt for a filename. The intended file name may be partially or fully typed (with auto-completion if partial), or an existing file may be selected to be overwritten. VE will prompt before overwriting any existing file. The file may be written to a directory other than the current directory by either typing in the partial or full pathname of the intended directory as part of the filename, or using the Open File Dialog to navigate to the directory of interest before entering the filename. Once a directory and file have been selected, this command writes the current file to that filename. This subcommand does not change the name of the current file, but merely writes a copy out under a different name.

20.3 Saving Your Work and Exiting

The Save subcommand of the Quit command (ESC-q/F12, s) updates the current file to disk and then exits VE (or switches to another open file, if more than one file is open).

20.4 Saving Your File via the Goto File Dialog

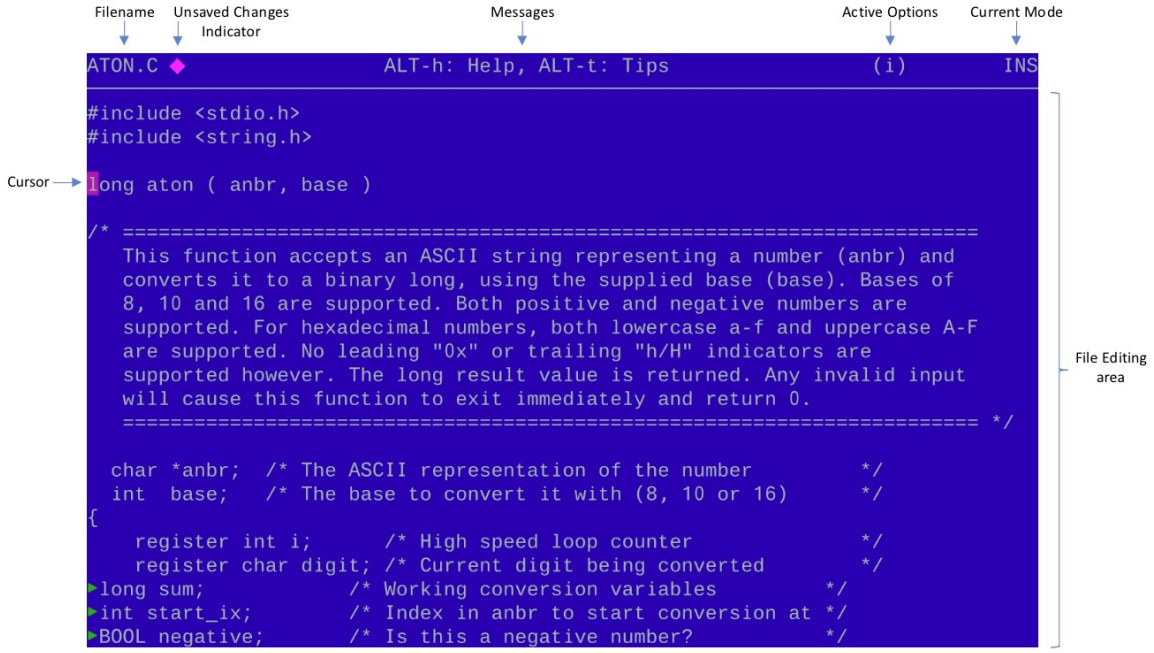
The later chapter in this document on working with multiple files presents another supported way of saving your work and exiting a file. This can be done via the Goto File dialog by selecting the open file of interest and typing CTL-s. See the later chapter on working with multiple files for full details.

20.5 Knowing When to Save - Change Tracking

VE monitors the files you are editing and knows when you have made changes to them since you opened them or saved them last. If you have unsaved changes, VE will display an Unsaved Changes indicator next to the file name on the Status Line.

The Unsaved Changes indicator is a colored diamond symbol that is displayed beside the file name on the Status Line any time there are unsaved changes to the file. The indicator appears whenever the first change is made to a file since it was opened or was last saved, and is removed whenever the file is saved or the changes themselves disappear (for example, one or more Undo commands reverse the changes).

The Unsaved Changes Indicator can be seen in the screen shot below, immediately to the right of the filename on the Status Line:




```
#include <stdio.h>
#include <string.h>

long aton ( anbr, base )

/* ==========================================================================
   This function accepts an ASCII string representing a number (anbr) and
   converts it to a binary long, using the supplied base (base). Bases of
   8, 10 and 16 are supported. Both positive and negative numbers are
   supported. For hexadecimal numbers, both lowercase a-f and uppercase A-F
   are supported. No leading "0x" or trailing "h/H" indicators are
   supported however. The long result value is returned. Any invalid input
   will cause this function to exit immediately and return 0.
   ========================================================================== */

  char *anbr;  /* The ASCII representation of the number         */
  int  base;   /* The base to convert it with (8, 10 or 16)      */
{
    register int i;      /* High speed loop counter              */
    register char digit; /* Current digit being converted        */
►long sum;            /* Working conversion variables         */
►int start_ix;        /* Index in anbr to start conversion at */
►BOOL negative;       /* Is this a negative number?           */
```

One of the key design principles of VE has always been blazing speed. VE has been written from “day one” in a very tight, efficient way. A tradeoff that has resulted from this is that the change tracking algorithm VE uses is not 100% perfect... 99.9% perfect the majority of the time, but not 100% perfect all the time. 99.9% is stated because there is one class of changes that the VE change tracking algorithm will not detect. If lines in a file are moved from one place to another, but are not changed, VE will not detect this as an unsaved change.

Even with this, change tracking is a CPU-intensive task. VE must examine the entire file after each command to detect any changes that the command may have made. This presents no observable performance impact on even the slowest machine VE has been verified on, a 10 MHz 286 under DOS 6.2. However, in the event that VE is run on a slower machine that does experience a performance impact as a result of change tracking, it can be disabled using the Options command. The Option's command BasePerf (Base Performance) subcommand allows users to enable or disable the Unsaved Changes Indicator. The complete keystroke is “ESC-o b u d”. This unpacks as [**O**ptions, **B**asePerf, **U**nsavedChanges, **D**isable].

21.0 Editing Multiple Files

21.1 Multiple File Support Overview

VE incorporates strong support for editing large numbers of files (up to 256) simultaneously. VE was conceived from "day 1" as a multi-file editor and this functionality is fully integrated throughout the program.

VE can manage up to 256 open files simultaneously. Both ve.exe/ve88.exe and ve32.exe support this number of files, although the practical realities of the 1M memory limit the 8086 addressing model imposes will almost certainly preclude simultaneous editing that many files with ve.exe/ve88.exe. Users who plan to edit large numbers of files at the same time, or edit small numbers of very large files, are advised to use ve32.exe, which can make use of PC extended/virtual memory.

VE designates one of the open files as the "current file" and displays that file on the screen. All other open files are open and available, but cannot be seen unless they are made the current file. This can be accomplished by any of a number of means, the simplest of which is the Next File command (ALT-+).

There is no need to update/save a file before switching to another open file. Moving to another file simply changes which of the files is displayed on the screen. It does not impact any other aspect of the file. VE allows users to switch freely between all available files, editing them all at the same time. If you think of the display window as simply a resource that can be attached to any open file, you have a fairly accurate view of how VE treats the association between the display window and the available file set.

21.2 Opening Multiple Files From the Command Line

VE can be started from the command line with multiple files, and after it is running, files can be freely added, discarded, and replaced.

To open multiple files from the command line at start up, simply put the files of interest on the command line that launches VE, per the below example:

ve file_1 file_2 file_3 ... file_n

Wildcards can be used if they are supported by the shell in use. For example:

ve *.c

will open all C source files in the current working directory. This will always work with ve32.exe, which provides built in file globbing. For ve.exe this will only work if the command interpreter expands wildcards as it passes command arguments to programs when it runs them. MS-DOS' command.com and Norton Utilities NDOS.com do not do this.

21.3 Opening Additional Files from Within VE

Once VE is running, additional files may be added at any time using the Edit command (ESC-e). The File and New subcommands of this menu will add an additional file to those already opened, and make that new file the current file (i.e. display it on screen).

The File subcommand ("f" subcommand of the Edit menu, also bound to the F8 key) presents the VE Open File Dialog to prompt for a filename and attempts to open the indicated file. If the file exists, it is opened, added to the current set of open files and made the current file. If it does not exist, an error message to that effect is posted to the messages area of the status line. The File subcommand fully supports auto-completion, so partial file or directory names can be typed and auto-completed. This process can be repeated until the filename of interest has been entered.

The New subcommand ("n" subcommand of the Edit Menu) opens a new unnamed file, adds it to the set of open files and makes it the current file (i.e. displays it on screen).

21.4 Discarding Files from Within VE

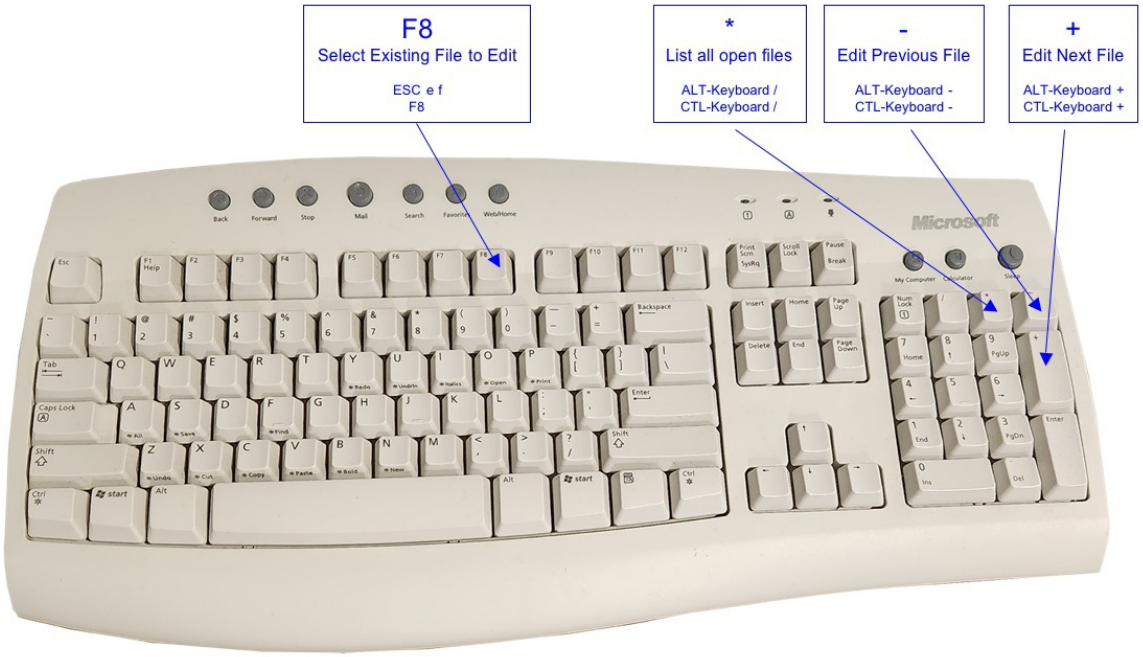
While VE is running, any currently open file may be discarded by using the [Quit, Discard] command (ESC-q, d). This command is also available as CTL-Keypad Del.

This command will prompt for confirmation if the file has been changed since it was opened or last saved. The Discard subcommand discards the current file and then quits VE itself, if no other files are open, or if there are other open files, moves to the next open file or presents the interactive Goto File dialog, which provides a list of available files and prompts for a file to move to. If only one other file is open, Discard simply moves to it. If more than one file is open, Discard may move to the last presented file, or may display a menu of presently open files and prompt for the file to move to.

21.5 Moving Between Open Files Sequentially

The Next File command (ESC-n, ALT-n, ALT-Keypad "+", CTL-Keypad "+", CTL-]) causes VE to move forward to the next open file, making it the current file and displaying it on screen. Repeated use of any of this command causes VE to cycle through all open files, wrapping around to the start when the last open file is reached.

In a similar way, the Previous File command (ALT-Keypad "-", CTL-Keypad "-" and CTL-[(must type CTL-[twice)) causes VE to move backwards to the last open file. In the current version of VE, there is no ESC or ALT equivalent of this command.



This functionality is also available via the Edit command's “+” and “-” subcommands. Note that as a convenience, this command supports use of the “=” command as well as “+”, since this is the unshifted version of the “+” key. The “-” key is already unshifted, and so no equivalent is needed for this key.

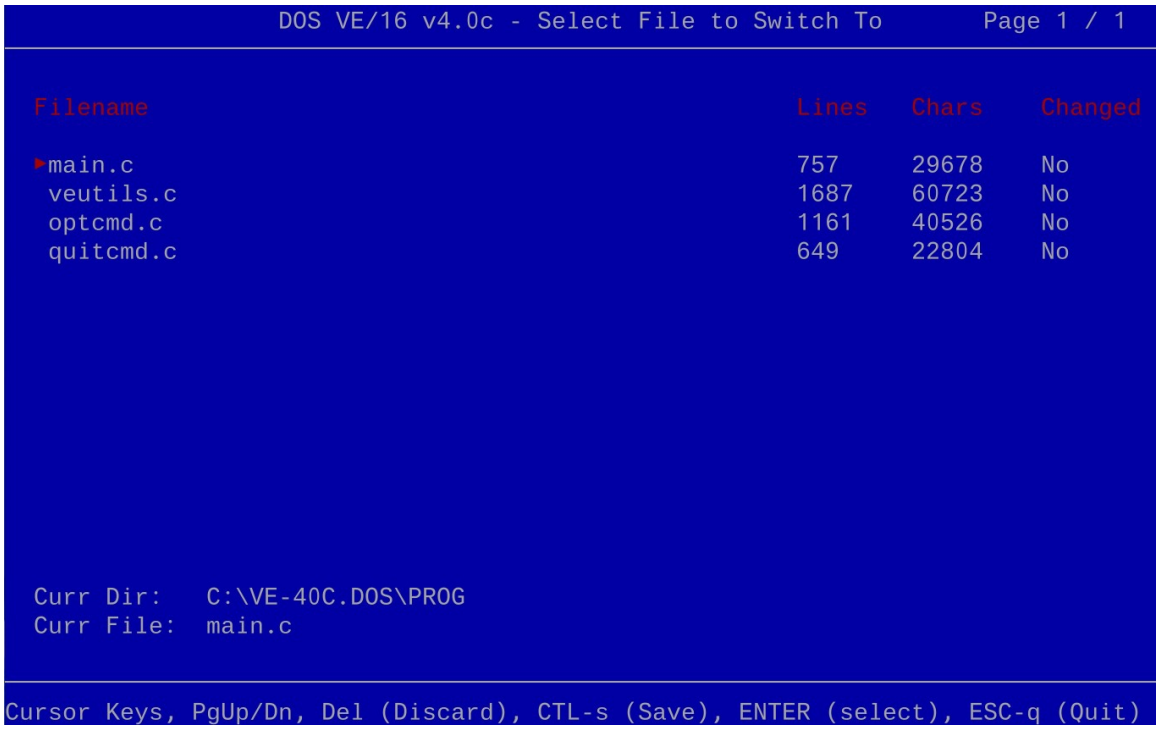
21.6 Inserting an External File into File Being Edited

While strictly not a multi-file topic, this seems the most appropriate place to document this capability. VE can read in an external file and insert it into the file currently being edited. Please see the [Block, Read] and [Column, Read] commands in the chapters “Working with Line Blocks” and “Working with Column Areas” respectively.

[Block, Read] inserts an external file into the file currently being edited as a series of lines. [Column, Read] inserts an external file into the file currently being edited as a column area (i.e. it can be inserted at an arbitrary column – [Block, Read] always inserts as full lines, starting at column 0).

21.7 Moving Between Open Files via Goto File Dialog

The Goto File command (ESC-g, ALT-g, ALT-Keypad “*”, CTL-Keypad “*”) cause VE to present its Goto File Dialog, an interactive full screen menu of all currently open files which allows users to select the next open file to move to.



This Dialog presents a number of information elements regarding each file:

- The file's name – the name of the file, potentially shortened to fit the available space
- The Unsaved Changes Indicator – this is presented immediately to the right of the file name if the file has unsaved changes and takes the form of the save colored diamond symbol used to reflect the same thing on the Status Line.
- The file's size in lines and chars
- The file's modification status – this is the textual equivalent of the Unsaved Changes Indicator (above) and is set to yes/no, reflecting whether there have been changes to the file since it was opened or last saved. This information appears in both the “Changed” column and as the Unsaved Changes Indicotr mentioned above.

Cursor up and down keys may be used to move the file selection carat to the file of interest, which may then be selected by pressing ENTER.

If the listing of currently open files exceeds one display page, the PgUp and PgDn keys may be used to move to the next/previous page of the listing as desired. As an alternate to using the PgUp or PgDn keys, the cursor up and cursor down keys may be used. When they reach the top or bottom of the displayed files, if there is another page, they will move forward to that page. In similar fashion, the PgUp and PgDn keys will wrap from the first page to the last, and visa versa.

As an alternate to interactively selecting a file of interest, the file's name may be partially or fully typed (with auto-completion if partially typed). It will be echo'd in the Curr File field, complete with full line editing support.

Users may cancel out of the Goto File Dialog at any time without selecting a file by pressing ESC-q. F12 achieves the same result.

Note that the cursor movement and PgUp, PgDn functions may also be selected by the same ESC-command_letter and keyboard shortcuts that apply to these keys while editing files in VE. Specifically, the following alternate forms are available while in the Goto File Dialog:

- Cursor Up - CTL-u, ESC-^
- Cursor Down - CTL-d, ESC-&
- Page Up - CTL-o, ESC-@
- Page Down - CTL-n, ESC-F

Finally, the mouse can be used in a “point and shoot” manner to select the file to move to. Activate the mouse pointer if it is not already on screen by left clicking the mouse or by typing the ALT-m (Mouse) command. Then move the mouse pointer to the file of interest and either single click, which will move the file selection carat to the filename that was clicked on, or double click, which will select the file and complete the dialog.

21.8 Saving and Discarding Files via Goto File Dialog

The Goto File dialog supports two additional functions that can be quite useful time savers. From within this dialog, users can update one or more files to disk and then discard them, or can discard one or more files without saving them, and can then select one of the remaining ones to move to.

To update a file to disk and then discard it, from the Goto File dialog simply move the file select carat to the file of interest and type CTL-s. The file will be saved and then discarded.

To discard a file without saving it, instead of CTL-s, type the keyboard Delete key. If the file has no unsaved changes, it will be discarded immediately. If the file has unsaved changes, VE will beep and present an informational message to that effect. The file will not be saved or discarded in this case. Note that users can tell if there are unsaved changes before attempting to discard a file file by observing the Unsaved Changes Indicator displayed immediately to the right of the filename on the dialog line if there are outstanding unsaved changes.

Via either CTL-s or Delete, if you discard the last open file, VE will simply quit and return back to the DOS command line interpreter.

22.0 Exiting from VE

22.1 Quitting VE from the Quit Command

VE's Quit command (ESC-q/ALT-q, F12) provides a comprehensive set of options for exiting VE, with or without saving the currently open file(s). The following Quit subcommands are available (in the Quit menu, simply type the first letter of the command):

- All / Absolute ("a" subcommand of the Quit menu) – this subcommand unconditionally exits VE, even if one or more of the currently open files has been modified since opened or saved last. If none of the open files has been changed since it was opened, VE simply quits with no further dialog. If one or more of the open files has been changed since it was opened, VE will prompt for confirmation, indicating the number of changed files. If confirmation is provided (type "y" to the prompt), VE discards all open files and quits.
- Discard / Quit ("d" and "q" subcommands of the Quit menu) – this subcommand discards the current file, and either moves to another open file, or quits VE if no other files are open. The Discard/Quit subcommand will prompt for confirmation if the current file has been modified since it was opened or since the last time it was saved. VE will then move to another open file, typically the file that was being edited before the file just discarded. This command is available via shortcut key "CTL-Keypad Del".
- Replace ("r" subcommand of the Quit menu) – this subcommand provides a convenient mechanism for chaining through a number of files, one by one. The Replace subcommand discards the current file, presents the VE Open File Dialog to prompt for a new filename, and then opens the selected file. In essence, it replaces the current file with a new file. If no file is selected (user cancels out of Open File Dialog) and no other files are open, VE simply quits. If other files are open, VE will move to another open file.
- Save ("s" subcommand of the Quit menu) – this subcommand saves the current file to disk and then either quits VE if no other files are open, or moves to another open file, typically the file that was being edited before the file just saved.
- Write ("w" subcommand of the Quit menu) – this subcommand writes the current file to disk with a different filename. It presents the VE Open File Dialog to prompt for a filename and then writes the current file to that filename. Files may be written to directories other than the current directory by either typing in the full or partial pathname (directory plus filename) with auto-completion or using the Open File Dialog to navigate to the directory of interest before entering the filename. Of course, the option exists to select an existing files to be overwritten. VE will prompt before overwriting any existing file. Note that the Write subcommand does not change the name of the current file, but merely writes out a copy of it under a different name.
- Update ("u" subcommand of the Quit menu) – this subcommand saves the currently editing file to disk and then returns to editing that file. As such, it does not quit VE at all, but is providing in the Quit menu as a convenience. [Quit, Update] is achieves the same result as CTL-s, F11 or ESC-U – it updates the file on disk and then returns to editing the file.

22.2 Quitting VE from the Goto File Dialog

Per the earlier section on saving/discarding files from within the Goto File dialog, if the only remaining open file is discarded, VE quits back to the command line interpreter that launched it.

23.0 VE Macros

23.1 VE Macros Overview

VE supports the definition and execution of keyboard macros. This is a capability that allows VE to learn keystrokes as they are entered and then play them back on command, as if they were being directly typed. This allows users to automate sequences of commands, repeating them at will. Once defined, macros can be saved as macro files and read back in to VE as needed, allowing them to be saved between editing sessions.

VE supports two kinds of macros: on-the-fly macros and regular macros. On-the-fly macros serve the purpose the name suggests – macros that are created with minimal overhead to help with some simple but repetitive editing task: quickly defined, used on the spot and then discarded. Regular macros are intended for more detailed sequences of editing actions where it would be laborious to have to re-enter the macro each time there was a need to use it, making the ability to save and reload it important. As a consequence, regular macros require a little more setup and must be given a name that can be used when the macro is saved to an external file for later reloading.

VE's on-the-fly macro capabilities are accessed via the CTL-a (AutoMacro) command. VE's regular macro capabilities are accessed via the obviously named Macro command (ESC-m). This command presents a submenu that contains commands to create macros, execute macros, save macros, load macros and list the currently available macros.

23.2 On-the-Fly Macros

To define and use an on-the-fly macro, simply type CTL-a. VE will enable the macro learning option, which will be visible from the display of the “m” indicator in the options area of the Status Line. While the macro learning option is enabled, VE will learn all of the keystrokes entered, recording them into the dynamic on-the-fly macro (named appropriately enough “auto”). To complete the learning phase, simply type CTL-a again. This action completes the on-the-fly macro, and the “m” indicator is removed from the options area of the Status Line.

To execute the newly created on-the-fly macro, position the cursor to the area of interest and enter the Macro Execute command, ESC-x or ALT-x. VE will execute the on-the-fly macro once and then stop. To repeat execution of the macro, use the F1 (Again) command or enter a repetition count prior to the Macro Execute command.

An important note about on-the-fly macros: they depend on no other macros being defined at the time they are used. If the only defined macro is the on-the-fly macro, VE knows to execute it when the Macro Execute command is entered. If there is more than one macro defined, VE must respond to the Macro Execute command by prompting for the name of the macro to execute. The on-the-fly macro, if defined, can still be executed when there is more than one macro defined, but VE has to first prompt for a macro name. To execute the on-the-fly macro, “auto” must be entered as the name of the macro to execute.

23.3 Creating a Regular Macro

Macros are created via the [Macro, Create] menu path (ESC-m, c). VE will prompt for a macro name, and will then enter macro definition mode. This is indicated by an “m” showing in the options area of the status line. Any desired name may be used for a macro, including “auto” – VE has no reserved macro names. Names may range from 1 character long to 20 characters long. If no name is entered (CR is typed in response to the name prompt) VE will assume the “auto” name and begin defining that macro. This is an alternate but slower way of defining the “on-the-fly” macro (see above). Is it therefore a “longcut” vs. a “shortcut”? :-)

In macro definition mode, VE learns all keystrokes as entered, recording them into the macro being created. When all of the desired keystrokes for the macro have been entered, the Macro definition is completed by returning to the Macro command and using the Stop menu item (ESC-m, s). The [Macro, Stop] command keys are not recorded with and do not form part of the macro.

23.4 Executing a Macro

To execute a macro, position the cursor at the desired starting point for the the macro and enter the [Macro Xecute] command (ESC-m, x). VE will prompt for a macro name, and then execute that macro.

If no macro name is provided (CR is typed in response to the macro name prompt) and there is only one macro defined, VE will select it and execute it. This is a convenient shortcut when working with only one macro.

23.5 Repeated Macro Execution

Macros may be repeatedly executed, via either repetition counts or via the Again command (F1 or ESC-a). To facilitate this, VE makes the [Macro, Xecute] command available as the free standing command ESC-x. This allows its use with repetition counts, and its repeated execution via the Again command. For example, “ESC 5 x” will prompt once for a macro name, and then execute that macro five times. “ESC-x” will prompt for and run a macro. The Again command will repeat that macro as often as wished, without reprompting for the macro name, including and repetition count that was part of the original macro Xecute command.

23.6 Undoing a Macro

VE fully supports undoing of the full sequence of steps a macro has executed via one Undo command. To undo the last macro executed, simply enter the Undo command (ESC-u or CTL-z). All steps taken by the macro will be undone in the correct order. This is true as well of extended sequences of editing steps, one or more of which are macros. Repeated use of the Undo command will undo each editing step in order. When the step to be undone is a macro, the entire macro will be undone and then the next editing step is available to undo.

23.7 Listing the Available Macros

The [Macro, List] command (ESC-m, l) will list all of the available macros in a full screen panel. At the bottom of the screen VE presents the usual full screen navigation guide, with ENTER as the only option. Pressing ENTER will return to editing the current file.

23.8 Deleting Macros

Any macros may be deleted with the [Macro, Delete] command (ESC-m, d). VE will prompt for the macro name and then delete the macro.

23.9 Saving Macros to Disk

Macros may be saved to disk files, for later reloading. Use the [Macro, Write] command to achieve this (ESC-m, w). VE will prompt for both the macro to write out and a file name to write it out under, and will then write the macro out to the indicated file. VE macro files are not text files and are not human readable or editable.

23.10 Loading Macros from Disk

Macros may be reloaded from disk files via the [Macro, Read] command (ESC-m, r). When entered, VE will prompt for a filename, and attempt to read a macro from that file. If the file exists, VE will load the macro and make it available for execution.

24.0 Miscellaneous Commands

24.1 Redrawing The Screen

In the unlikely event that VE should encounter a bug that results in the screen image not matching the contents of the file being edited, the screen image can be redrawn via the View command (ESC-v/ALT-v).

The author is not aware of any such bugs in v4.4a, but if you should encounter one, you will want to know that there is a **$US 10.00 "bounty"** on each one. If you are the first person to report a previously unknown screen mismatch bug, the author will send you a crisp new $US 10.00 bill (or the equivalent in your local currency).

To qualify, the screen corruption bug must be fully reproduce-able. Email support@inverary.net with the title "VE Screen Mismatch Bug" and outline the steps which will reliably reproduce the screen mismatch. Please include as an attachment the file being edited (the ability to reproduce a screen mismatch bug has been notoriously dependent on the content being edited). Indicate the line number and cursor column position VE must be at so that the sequence of steps you provide results in a visible screen mismatch. There is no limit to the number of bounties any one person can collect! Happy Hunting!

24.2 Changing Case

The Change Case (ESC-~, ALT-~) command changes the case of the character the cursor is on, either from lower to upper or from upper to lower. The command is direction sensitive. After the current character's case has been changed, the cursor will move either to the left or to the right, dependent on the last cursor motion command. For example, Cursor Right followed by Change Case will change the case of the character under the cursor and then advance the cursor one position to the right. Similarly, Cursor Left followed by Change Case will change the case of the character the cursor is on and then position the cursor one position to the left.

Conveniently therefore, the case of runs of consecutive characters can be changed in sequence by simply repeatedly using the Again (F1) command. This will proceed to the left or to the right, depending on the last cursor movement command.

Note that there is no Undo for Change Case, since the command itself is its own undo!

24.3 About VE

The About VE command (ESC-A) provides brief information on the origins and history of the VE text editor. The same information is available in the opening pages of this manual.

This information is presented as a full screen panel. At the bottom of the screen the usual full screen navigation guide is presented, with ENTER as the only option. Pressing ENTER will return to editing the current file.

24.4 Shell to DOS

VE supports shelling out to DOS, performing DOS commands and then returning to VE. To shell out to the DOS command line, enter the ALT-d (DOS) command.

Note that the ability to execute this command successfully is heavily dependent on there being enough free memory to support it. When editing large files, or multiple files, this command may gracefully fail due to insufficient available memory.

This command looks for and will use the command line specified by the COMSPEC environment variable. If that variable is not set, it will look for 4DOS.COM, NDOS.COM and COMMAND.COM in that order. In the event that none of these are available and COMSPEC is not set, the command will fail.

24.5 A Look At VE's Internal Status

For the curious, VE provides the Status command (ESC-s/ALT-s), which displays selected key details of VEs current internal status. This command lists multiple information points:

- Current filename
- Current file size, number of lines
- Current file format
- Line Buffer size, number of lines
- Block Buffer size, number of lines
- Column Buffer size, number of lines
- Total amount of memory in use for files and buffers
- Status of the Undo System
- Number of files currently open
- Number of macros defined
- A set of graphics-related information, including text resolution, video mode and more
- A set of mouse-related information
- Key auto-detected system information: Graphics Type, CPU Type

This information is presented as a full screen panel. At the bottom of the screen the usual full screen navigation guide is presented, with ENTER, q and Q as the available options. Pressing any of these will return to editing the current file.

25.0 Configuring VE

Numerous aspects of VE's presentation and behavior can be modified using the Options command (ESC-o, ALT-o). The Options command can be used to:

- Change VE's color schema
- Change parameters related to TAB handling
- Change parameters that affect VE's CPU consumption
- Enable/disable auto indentation
- Change the characters used for start-of-line / end-of-line matching
- Enable/disable on-the-fly text wrapping (“power typing”)
- Set the file format that VE will use (DOS or Unix)

Each of these options is described below.

25.1 Configuring VE's Color Schema

VE defaults to using a color schema of white on blue. However, these colors may be changed via the [Options, Colors] menu path (ESC-o, c).

The [Options, Colors] command presents a submenu which allows you to set the foreground (text) and background colors. As you change these colors, they become the “working set” of colors, but have not yet been applied to the display. The Apply subcommand applies the colors to the display, making them the displayed screen colors. As with all VE options, the selected colors are saved to the file $HOME\ve.opt when the [Options, Save] command is entered (ESC-o, s). Subsequent starts of VE will load this file and change the colors to those specified in the it. Hence, to “permanently” change the colors used by your version of VE, change the colors to suit your taste, and then save your options. Henceforth, VE will use the new colors.

The [Options, Colors] menu selection presents the following choices – type the first letter of any command to select it:

- Palette – selecting Palette displays the available colors on the status line. As of VE 4.0, VE supports all sixteen standard VGA colors. Unlike most things in VE, color names must be typed in their full format. This is due to overlaps in the first letter of several color names.
- Working – selecting Working shows the names of the current working set of foreground and background colors.
- Foreground – selecting Foreground prompts you to select a new foreground color for the working set from the set of colors that are available (see Palette). The current working foreground color is included as part of the prompt. To keep the current foreground color, press ENTER. To change it, enter an available color name and press ENTER. The new color becomes the working foreground color, but is not yet applied to the display.
- Background – like Foreground, but manipulates the working background color.
- Cursor – selecting Cursor prompts you to enter a new color from the set of available colors for the text cursor (see Palette). To keep the current color, press ENTER. To change it, enter an available color name and press ENTER. The new color becomes the working cursor color but is not yet applied to the display.
- Apply – Selecting Apply applies the current working colors to the display, making them the displayed colors.

25.2 Configuring Auto Indent

With its default settings, VE auto indents. This is visible via the display of the “i” indicator in the opions area of the Status Line. With auto-indent active, when ENTER is pressed at the end of a line of text, VE automatically indents the next line to the same level of indentation as the line just typed. This behavior may be enabled or disabled via the [Options, Indent] menu path (ESC-o, i). Here, it is possible to select “Enable” or “Disable” to control this function.

25.3 Configuring Tabs

Tabs may be configured via the [Options, Tabs] menu path (ALT-o, t). The Tabs submenu supports enabling and disabling of input and output tab translation, and also allows modification of the tabstop value that will be used.

Input tab translation will expand all tabs encountered as a file is read in, expanding them to spaces according to the current tab stop. It will also expand any TAB characters that are typed as content into a file being edited. The default value for Tabstop is 4. This can be changed via the [Options, Tabs, Tabstop] menu path (ESC-o, t, t). Input tab translation is OFF by default. When input tab translation is off, tabs in the file content are displayed as (typically) green carat symbols.

Please note that when working with “make” files, input and output tab translation should be off before editing the makefile. This is because most “make” implementations require the presence of TAB characters as part of their specific syntax.

Output tab translation compacts all possible white space into tabs before writing the file, again using the current tab stop. In keeping with the general idea of compacting the size of the file, output tab translation also strips any trailing white space from the end of lines, to minimize the size of the file.

Tab translations only effect the file being edited on input and output. Turning input translation on after loading a file containing tabs without input translation on will not expand the tabs on the fly. Similarly, turning output tab translation on and writing the file out does not compact the currently editing version of the file, but only the version written to disk.

25.4 Configuring Find/Replace's Start/End Match Characters

As outlined earlier in this document, VE supports two special characters that match the start of a line and the end of a line. These characters can be used in Find/Replace strings to match the start or end of a line, or words that occur at the start or end of a line. With its default settings, VE uses the “^” character to match the start of a line, and the “$” to match the end of a line. These characters may be changed with the [Options, Match] menu path (ESC-o, m). Here, using the Start-of-Line and End-of-Line menu selections (type “s” or “e”), it is possible to change the character used. Be very careful with your choices if you should elect to change the default match characters!

Note that changing these characters also changes the characters used by the Open File and Goto File dialogs' auto-completion feature for denoting filenames that “start with” and “end with” entered partial file/directory names.

25.5 Configuring Text Wrapping

See the earlier section of this document on Formatting Text for full details regarding the use and configuration of text wrapping.

25.6 Configuring Text File Format

As outlined earlier in this document, VE fully supports both DOS/Win and Linux/MacOSX formatted text files (Linux/MacOSX uses a single Line Feed character as a line end; DOS/Win uses both Carriage Return and Line Feed). By default, VE determines the text format of a file as it is read in, and uses the same format when it is written out.

The format used to write any given file out can be modified while the file is open by using the [Options, Fileformat] command (ESC-o, f). Here it is possible to select “dos” or “linux” (type “d” or “l”). The selected format will be used to write out the current file. However, this is not a global setting. VE's default behaviour is always to write any given file in the format it was read in. If the current file format is changed with the [Options, Fileformat] menu path, that change is only applicable to the file VE is editing on screen at the time. All other files will continue to be written in the format that they were read in.

25.7 Configuring VE CPU Performance Consumption

VE has been written from “day 1” for tight, fast code. In VE 4.x, somewhat at variance with this general design mantra, two moderately CPU-hungry features were added: (1) Unsaved Changes Indicator and (2) Software Cursor. However, in keeping with the “tight” and “fast” design goals, options have been added to enable/disable both of these features, so that they can be turned off if VE is being run on a low powered machine (typically an 8088-based PC or PC XT).

The Unsaved Changes Indicator feature provides a visual indication on the status line if a file has unsaved changes. This indicator comes on whenever unsaved changes exist and goes off whenever the file is saved to disk or the unsaved changes are undone, typically by an Undo command. This feature is moderately CPU-hungry because it must scan at least the current screen, and often the entire file, after each command is executed to determine if any changes have occurred to the file.

The Software Cursor feature replaces the blinking hardware cursor with a more visually pleasing non-blinking solid block cursor whose color can be configured via the Options command. This feature is also moderately CPU-hungry because on a character by character basis, it must track the exact position of the hardware cursor and replace it with an overlay.

To enable/disable of these two features, VE 4.0 introduced a new Option subcommand, BasePerf (Base Performance). This command is available via the (ESC-o, b) menu path. This command presents the current status of the Software Cursor and the Unsaved Changes Indicator and allow users to select either one: “s” for Software Cursor and “u” for Unsaved Changes Indicator. Once selected, users may Enable or Disable the selected feature.

The slowest machine VE has been tested on is a 4.77 MHz 8088-based PC XT, and there is no observable performance impact from either of these features on that machine. Despite this, if VE seems to lag on the machine you are running it on, disabling one or both of the above two features should return VE to its intended snappy performance.

25.8 Saving The VE Options Configuration

The VE options configuration may be saved at any time via the [Options, Save] command (ESC-o, s). This writes the current options to the file “ve.opt” (or ve32.opt for ve32.exe) in the current HOME directory. When VE starts up, it looks for the ve.opt/ve32.opt file. If found, it will load its options from there. If not found, it uses its internal defaults. Hence, VE can be “permanently” customized by setting options as desired, and then saving those options via the [Options, Save] menu path. VE will use those options on all subsequent starts. This functionality is only active if the HOME environment variable is set.

Note the since ve.exe and ve32.exe save their options in separate files, it is possible to setup these two versions of VE differently, allowing for use with different option sets as desired.

25.9 Loading The VE Options Configuration

VE will autoload its options from the file $HOME\ve.opt (or $HOME\ve32.opt for ve32.exe) each time it starts. Hence, if options have been previously saved via the Option command's Save menu item, those options will be automatically restored each time you start VE. This functionality is only active if the HOME environment variable is set.

26.0 Miscellany

26.1 Repetition Counts Default Commands to Unmodified Form

The meaning of several PC keyboard keys can be modified by pressing ESC key and then the key itself. An example of such keys are the cursor left and cursor right keys. Entered by themselves, they move one character to the left or the right. Preceded by ESC, they are modified to move one **word** to the left or right.

While use of a preceding ESC to modify the meaning of a command key is a simple and easily understand notation, it creates a logical dilemma when a repetition count is entered, since repetition counts must be preceded by an ESC to move into Command mode first. So, if “ESC 5 cursor_right” is entered, does it mean “move 5 chars to the right” or “move 5 words to the right”? VE resolves this dilemma by assuming that the ESC is logically part of the repetition count, not part of the command, and thus defaults the command to its unmodified form. In the example above, VE defaults the command to “move 5 chars” not “move 5 words”.

As a practical consequence of this, it is not possible to enter a repetition count with any of the modified commands available via ESC and such keyboard keys. To enter a repetition count with such a command, it is necessary to use the “ESC-letter” form of the command, rather than the simple keyboard key form. In the example above, to move to the right five words, it is necessary to enter “ESC 5 ESC-]”.

The keyboard keys in this category include:

- Cursor Left: char left or word left
- Cursor Right: char right or word right
- Cursor Up: line up or paragraph up
- Cursor Down: line down or paragraph down
- Page Up: page up or move to top of file
- Page Down: page down or move to end of file
- Delete: delete char or delete word

26.2 The “*” Repetition Count Isn't Actually Infinity

Previously in this document, it has been noted that to replace all occurrences of something using the replace command, a repetition count of “*” should be used. This remains true. However, it is worth noting that “*” is actually translated by VE into the “very large number” 32,000 (near the safe limit of an unsigned 16 bit integer, the lowest common denominator for integer sizes). If a “*” repetition count is used, and there are more than 32,000 occurrences to replace, it may be necessary to repeat the replace operation (use the Again command (ESC-a or F1).

26.3 VE File Writes Overwrite “filename~”

To protect users from potential file content loss, whenever VE writes out a file, it first writes to a temporary file, and then if that write succeeds, deletes the original and renames the temporary to the original's filename. This approach protects users in the unlikely event that a bug in VE's file writing code results in total loss of file contents.

VE constructs the temporary file name by overwriting the last character of the current file's name with the character “~”. A consequence of this is that there is a pre-existing file whose name matches the name of the file being written, but with “~” as the last character, that file will be effectively deleted (it is overwritten with the contents of the current file being written out, and then renamed to the name of that file). This is intended operation and not a bug!

Users of VE are advised to avoid use of the character “~” as the last character of filenames in directories where they may be using VE!

27.0 Known Issues

There are several known issues with VE v4.4a. CampbellWare is aware of these issues and working on them. There always has to be SOMETHING for the next release! Please do not report these as bugs.

The issues are:

27.1 Open/Goto File Dialogs and Macros

VE does not support the use of the Open or Goto File Dialogs from within a macro. There is no explicit interlock to stop the use of these dialogs however, and attempts to use them from within a macro will fail in all sorts of horrible and messy ways!

27.2 Auto Indent and On-the-fly Text Wrapping

These two options are mutually exclusive in VE. If Auto Indent is enabled via the Options command, On-the-fly Text Wrapping is automatically disabled. The inverse is true as well: enabling On-the-fly Text Wrapping disables Auto Indent.

27.3 On-the-fly Text Wrapping and Justify/Delete

On-the-fly Text Wrapping can appear to get “stuck” when deleting or rubbing out spaces, if the Right Justify option has been set. This is because in some situations, VE's right justify function may immediately replace the space just deleted, resulting in no visible change to the file. When this occurs, repeated delete/rubouts of spaces under the cursor will appear to not occur, and the text wrapping function will seem to be stuck. There are two simple solutions to this problem:

- Cursor over to a non space character and carry on
- Disable the Right Justify functionality

27.4 Support for Text Files with Lines >384 Chars

VE supports line sizes up to 384 chars. This is an arbitrary size selected by the author as being sufficient for the majority of text files. It is important to understand how VE handles files whose lines exceed 384 characters.

As files with lines longer than 384 characters are read in, the lines longer than this value are broken into multiple separate lines, each one of which is no longer than 384 characters, including the trailing line end characters (LF for Linux/MacOSX formatted files, and CR, LF for DOS/Win formatted files). This allows the files to be viewed, even if they cannot be actively edited. It is imperative that no attempt to be made to write such files back out. VE **will** write the file back out, with the lines broken as described above. This will destroy the integrity of the original file.

27.5 Screen/File Mismatch

VE v4.4a has no known issues that result in the contents of the displayed screen not matching the contents of the file being edited. However, if you should encounter such an issue, you will want to know that there is a **$US 10.00 "bounty"** on each one. If you are the first person to report a previously unknown screen mismatch bug, the author will send you a crisp new $US 10.00 bill (or the equivalent in your local currency).

To qualify, the screen mismatch bug must be fully reproduce-able. Email support@inverary.net with the title "VE Screen Mismatch Bug" and outline the steps which will reliably reproduce the screen mismatch. Please include as an attachment the file being edited (the ability to reproduce a screen mismatch bug has been notoriously dependent on the content being edited). Indicate the line number and cursor column position VE must be at so that the sequence of steps you provide results in a visible screen mismatch. There is no limit to the number of bounties one person can collect! Happy Hunting!

Happily, screen mismatch bugs are recoverable by user action. In the unlikely event that VE should encounter such a bug, the correct screen image can be redrawn via the View command (ESC-v/ALT-v).

28.0 Bug Reports, Donations, Gratuitous Praise

Please send bug reports, donations, gratuitous praise and any other form of communication with the author of VE by emailing:

support@inverary.net

CampbellWare will make every effort to respond to all such emails.

29.0 Appendix A – VE Keyboard Commands

29.1 About VE Keyboard Commands

VE assumes the presence of the now standard 101 key PC Extended Keyboard. The command key sequences documented in this section have been verified on the PC Extended Keyboard only, and not all command sequences may work as documented on other keyboards, such as the 82 and 96 key keyboards that were sold with the PC and PC XT respectively.

29.2 PC Extended Keyboard Key Commands

Key Binding	Command
Insert	Insert/Overwrite
Delete	Delete Char
Rubout	Rubout
Home	Start of Line
End	End of Line
Page Up	Page Up
Page Down	Page Down
Cursor Up	Cursor Up
Curor Down	Cursor Down
Cursor Right	Cursor Right
Cursor Left	Cursor Left
Keypad 5	Move to Screen Edge (per last cursor motion)
Keypad Ins	Insert/Overwrite
Keypad Del	Delete char

29.3 Function Key Commands

Key Binding	Command
F1	Again
F2	Block
F3	Column/Rectangle
F4	Jump
F5	Find
F6	Replace/Substitute
F7	Paste/Get
F8	Edit – Open Files for Editing
F9	Delete Line
F10	Delete Right
F11	Update
F12	Quit

29.4 ESC Key Commands

Key Binding	Command
ESC Insert	Insert/Delete
ESC Delete	Delete Word
ESC Page Up	Top of File
ESC Page Down	Bottom of File
ESC Home	Cursor Left of Screen
ESC End	Cursor Right of Screen
ESC Cursor Down	Paragraph Down
ESC Cursor Up	Paragraph Up
ESC Cursor Left	Word Left
ESC Cursor Right	Word Right
ESC Keypad -	Half Screen Up
ESC Keypad +	Half Screen Down
ESC 0	Start of Line
ESC Dollar	End of Line
ESC a	Again
ESC b	Block
ESC c	Column/Rectangle
ESC d	Delete Left
ESC e	Edit
ESC f	Find
ESC g	Goto File
ESC h	Help
ESC i	Info
ESC j	Jump
ESC k	Delete Right
ESC l	Delete Line
ESC m	Macro
ESC n	Next File
ESC o	Option
ESC p	Paste/Get
ESC q	Quit
ESC r	Replace/Substitute
ESC s	Status
ESC t	Tags
ESC u	Undo
ESC v	View
ESC w	Wrap Paragraph
ESC x	eXecute Macro
ESC y	Copy Line
ESC z	Pick (Move line to top column to left)

ESC A	About VE
ESC F	Page Down
ESC I	Insert/Overwrite
ESC O	Insert/Overwrite
ESC E	Encode File
ESC D	Decode File
ESC W	Delete Word
ESC C	Copy Line
ESC X	Cut Line
ESC V	Paste Line
ESC T	Cursor Top of Page
ESC G	Cursor Middle of Page
ESC B	Cursor Bottom of Page
ESC H	Cursor Top of Page
ESC M	Cursor Middle of Page
ESC L	Cursor Bottom of Page
ESC Q	Cursor Left of Screen
ESC Y	Cursor Middle of Screen
ESC P	Cursor Right of Screen
ESC U	Update
ESC J	Null
ESC K	Null
ESC L	Null
ESC N	Null
ESC P	Null
ESC S	Segment Status
ESC Z	Bottom of File
ESC ,	Cursor Left
ESC .	Cursor Right
ESC ^	Cursor Up
ESC &	Cursor Down
ESC ;	Extended Motion (Page Up Page Down Home End)
ESC {	Word Left
ESC }	Word Right
ESC [	Para Up
ESC]	Para Down
ESC ~	Change Case
ESC `	Change Case
ESC :	Move to Screen Edge (per last cursor motion)
ALT _	Null
ESC #	Rubout

```
ESC <               Start of Line
ESC >               End of Line
ESC @               Page Up
ESC /               Top of File

ESC -               Half Screen Up
ESC +               Half Screen Down
ESC (               Half Screen Left
ESC )               Half Screen Right
```

29.5 ALT Key Commands

Key Binding	Command
ALT Insert	Null
ALT Delete	Delete Word
ALT Home	Top of File
ALT End	Bottom of File
ALT Page Up	Half Screen Up
ALT Page Down	Half Screen Down
ALT Cursor Up	Paragraph Up
ALT Cursor Down	Paragraph Down
ALT Cursor Right	Half Screen Right
ALT Cursor Left	Half Screen Left
ALT Keypad Ins	Insert
ALT Keypad Del	Delete Word
ALT Keypad /	Pick (line to top column to left)
ALT Keypad *	Goto File
ALT Keypad -	Prev File
ALT Keypad +	Next File
ALT Rubout	Rubout
ALT Keypad Enter	Null
ALT a	Auto Macro
ALT b	Block
ALT c	Column/Rectangle
ALT d	Shell
ALT e	Edit
ALT f	Find
ALT g	Goto File
ALT h	Help
ALT i	Info
ALT j	Jump
ALT k	Delete Right
ALT l	Delete Line
ALT m	Toggle Mouse
ALT n	Next File
ALT o	Option
ALT p	Paste/Get
ALT q	Quit
ALT r	Replace/Substitute
ALT s	Status
ALT t	Tips
ALT u	Undo
ALT v	View
ALT w	Wrap Paragraph
ALT x	eXecute Macro
ALT y	Null
ALT z	Pick (line to top column to left)

```
ALT [               Word Left
ALT ]               Word Right
ALT /               Top of File
ALT ~               Change Case
ALT `               Change Case

ALT ;               Extended Motion (Page Up Page Down Home End)
ALT '               Top of File
ALT ,               Cursor Left
ALT .               Cursor Right
ALT /               Null
ALT -               Half Screen Up
ALT +               Half Screen Down
```

29.6 CTL Key Commands

Key Binding	Command
CTL t	Cursor Top of Page
CTL g	Cursor Middle of Page
CTL b	Cursor Bottom of Page
CTL q	Cursor Left of Screen
CTL y	Cursor Middle of Screen
CTL p	Cursor Right of Screen
CTL u	Cursor Up
CTL d	Cursor Down
CTL l	Cursor Left
CTL r	Cursor Right
CTL e	Extended Motion (Page Up Page Down Home End)
CTL n	Page Down
CTL o	Page Up
CTL c	Copy Line
CTL x	Cut Line
CTL a	Append Line (copy append to Line Buffer)
CTL v	Paste
CTL s	Update
CTL z	Undo
CTL h	Rubout
CTL j	Word Left
CTL k	End of Word
CTL f	Null
CTL i	Null
CTL m	Null
CTL w	Null
CTL Keypad +	Next Open File
CTL Keypad -	Previous Open File
CTL Keypad *	List/Select Open Files
CTL Keypad Del	Discard current file, quit VE if it is only file
CTL]	Next Open File
CTL [	Previous Open File (must type twice)
CTL PgDn	Move a half page down
CTL PgUp	Move a half page up
CTL Tab	Null
CTL Enter	Null

30.0 Appendix B – A Brief VE Release History

30.1 Release History

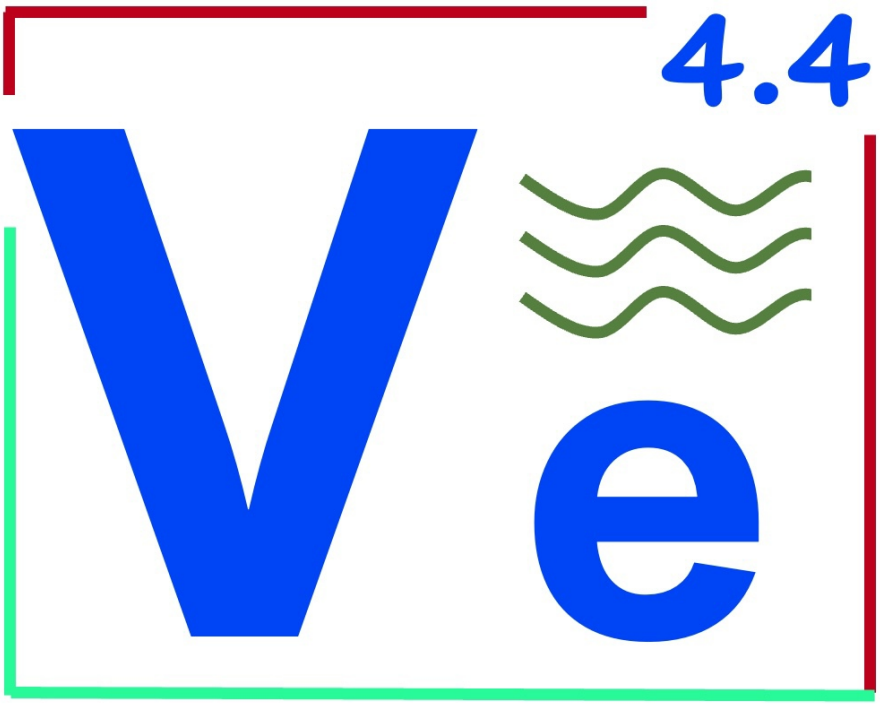
The following is a brief VE release history:

- v1.x – 1984. Initial version of VE, released on Qunix (a unix lookalike) for the NABU 1600 small business computer (8086, 256KB/512KB RAM). Basic text editing functionality, macros and on-the-fly text wrapping (“Power Typing”).
- v2.x – 1994. Release of VE for MS-DOS and the x86 PC. Port of v1.x from Qunix to MS-DOS, addition of word and paragraph support and after-the-fact text wrapping.
- v3.0 – 2004 – Initial release of VE for Linux. Significant number of new features added, including support for Tags, Help and About commands, configurable screen colors and much more. The initial version of the interactive Goto File Dialog debuted in this release. Numerous bugs fixes from v1.x and v2.x were included.
- v3.1 – 2004 - Added Global Undo
- v3.2 – 2004 - Added screen scrolling left and right
- v3.3 – 2005 - Added Column/Rectangle support
- v3.4 – 2005 - Added Mouse support (Linux only)
- v3.5 – 2005 - Added full-screen, interactive Open File Dialog, rewrite of Goto File Dialog to match, mouse support for both of these dialogs (Linux only) and global line editing for all VE prompts.
- V4.x – 2022/23 – Released for DOS only. Added MDA/Hercules, CGA and EGA support to the existing VGA support, 80x50 and 80x43 screen resolutions, PC, PC XT and PC AT support, DOS mouse support, Unsaved Changes Indicator, Software Cursor, Filename Auto-completion, Intelligent Paste, line, block and column Append commands, expanded Help command, on-demand tips and numerous performance enhancements and bug fixes. Updated the VE User Manual (this document) and added new Quick Start Guide and Quick Reference Guide documents. See below for the full list of VE 4.x changes.

30.2 New in Release 4.x

The following content is new in release 4.0c (November 2022) / 4.2e (April 2023):

- MDA/Hercules (80x25), CGA (80x25) and EGA support (80x25, 80x43), available via startup "-s" option – VGA (80x25, 80x43, 80x50) supported from release 1.x onwards
- Automatic detection of graphics type
- Manual selection of graphics type, available via "-g" startup option; allows manual override of above automatic detection of graphics type
- PC, PC XT and PC AT support, including additional key bindings to support the now standard 101 key Enhanced PC Keyboard on PC and PC XT.
- ve88.exe build, targeting PC and PC XT machines/clones (8086, 8088)
- DOS Mouse support
- Unsaved Changes Indicator
- Software Cursor
- Filename and directory name auto-completion
- Intelligent Paste
- On-the-Fly Macros
- Change Case command
- Shell to DOS
- Line, Block and Column/Rectangle Append subcommands
- Expanded Help screens
- On-demand tips
- Horizontal Pick (move column to left screen edge)
- Cursor motion commands to move to Top, Middle and Bottom of display
- Cursor motion commands to move to Start, Center and End of line
- Additional vi-like key bindings (yank/put, change case, ESC-0, ESC-$, etc.)
- CTL-s key binding to Update command; CTL-s now saves the current file
- Near ubiquitous use of CTL-c, CTL-x and CTL-v to copy, cut and paste lines, blocks and rectangles
- Complete rewrite/optimization of VE's screen display layer
- Complete rewrite/optimization of VE' s command dispatch layer
- Complete rewrite/optimization of VE's base string moving/setting primitives (ve.exe, ve88.exe)
- Change of TAB display character to (typically) green carat symbol
- Change of Block and Column selection markers to (typically) red carat symbols
- Disable of NUMLOCK when VE starts, restoration of its original state when VE exits
- Multiple bug fixes from earlier versions of VE



End of Document