VEDIT
Customizable Full Screen Editor

CompuView

VEDIT 1.39/1.17

Full Screen Editor

User's Manual

Written By

Theodore Green

CompuView Products, Inc.

1955 Pauline Blvd.
Ann Arbor, MI 48103

Copyright (C) 1980, 1984 by Compuview Products, Inc. All rights reserved worldwide. No part of this publication may be reproduced, in any form or by any means, for any purpose without the express written permission of Compuview.

DISCLAIMER

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

Last Manual Revision: November 13, 1984

Table of Contents

I. Introduction 5

Getting Started 7
Installation 7
Using this Manual 7
Sample Edit Session 9
Overall Description 13
Basic Editing Concepts 13
Visual Mode 15
Command Mode 17
Which Mode to use for What 19
Word Processing with VEDIT 20

II. Tutorial 21

Invoking VEDIT 23
Keyboard Characters & Edit Functions 25
Cursor Movement 26
Page Movement 28
Entering New Text 29
Visual Functions 31
Deleting Text 31
Correcting Mistakes Made to a Line 33
Repeating Operations 34
Indenting Text 35
Word Wrap and Formatting Paragraphs 37
Moving and Copying Blocks of Text 38
Emptying a Text Register 40
Sending Text to the Printer 41
Entering Control Characters in the Text 41
Searching 42
Replacing 43
Switching Between Visual and Command Mode 44

Saving Already Edited Text 45
Begin Editing New File 45
Making More Memory Space 46
Disk Directory Display 46
Inserting a Line Range of another File 47
Concatenating Two Files 48
Splitting a File into Two or More Files 49
Recovery from Full Disk Errors 50
Ending the Edit Session 51

III. Memory and File Management 53

Text Registers 54
Automatic Disk Buffering 55
Forward Disk Buffering 56
Backward Disk Buffering 57
Automatic Startup 58
Global File Operations 59
Multi-tasking Operating Systems 59

IV. Visual Mode 61

Screen Display 62
Status Line 62
Keyboard Characters 64
Entering New Text 64
Performing Edit Functions 65
The Repeat Function 65
Horizontal Scrolling 66
Cursor Movement 66
Setting and Jumping to Text Markers 67
The Tab Character 67
Search and Replace 68
Cancel Function 70
The Text Registers 70
Indent and Undent Functions 71
Printing Text 72
Word Processing Functions 72
Word Wrap and Margins 73
Formatting Paragraphs 73
Lower and Upper Case Conversion 74
End of Lines 75
Inserting Control Characters 75
High Bit Character Support 75
Disk Buffering in Visual Mode 77
Edit Functions 78

V. Command Mode 83

Command Mode Notation 84
Command Lines 84
Command Line Editing 85
Help Command 85
Controlling Console Display 86
File Operations 87
Exiting Without Saving 87
Quitting Without Saving 87
Save File and Continue Editing 87
Editing a Second File 88
Directory Display 88
Extracting Portions of Other Files 89
Disk Buffering in Command Mode 89
Disk Write Error Recovery 90
Command Mode Features 93
Search Options 93
Pattern Matching 95
Iteration Macros 97
Printing Text 100
WordStar (TM) Files 100
Text Registers 101
Command Macros 103
Additional Command Macro Features 105
Re-routing Console Output 106
Numerical Capability 106
Print Formatter Command Macro 107
Brief Command Summary 108
Detailed Command Descriptions 114

VI. APPENDICES 151

A - Customizing VEDIT 153

What is Customization 154
When is Customization Necessary 154
CRT Terminal and Memory Mapped 155
How to Perform Customization 156
Quick Customization 172
Customization Notes 172
VEDIT Checksum 172
Keyboard Layout 172
A Word About Keyboards 174
Screen Size Parameters 174
Memory Size Parameters 175

B - Quick Command Reference 177

C - Error Messages 181

D - General Versions 185

Description of files on disk 186
Personal Keyboard Layout 187
Default CRT Customization 188
Default Keyboard Layout 189
HEATH H-19 190
IBM 3101 192
TVI 920C 193
TVI 950 194

E - 8086 Versions - Specific Screens 195

PCDOS and TANDY 2000 Files 196
CP/M-86 Files 197
IBM PC Keyboard Layout (PCDOS-CP/M-86) 198
IBM PC Keyboard Layout Concurrent CP/M 199
TANDY 2000 201
ZENITH Z-100 203
IBM Displaywriter 205
NEC APC 208
VICTOR 9000 209
TI Professional 212

F - 8080/Z80 Memory Mapped Screens 213

TRS-80 Mod I 214
TRS-80 Mod II 217
IMSAI VIOC 221

G - 8080/Z80 Versions - Specific Screens 223

Superbrain 224
Apple II 227
Xerox 820 230
Osborne I 231

I N T R O D U C T I O N T O V E D I T

Introduction to VEDIT

VEDIT is an editor designed to take full advantage of a CRT display to make your word processing and program development editing as fast and easy as possible. VEDIT's Visual Mode offers true "What you see is what you get" type editing, which continuously displays a region of your file on the screen and allows any changes made to the screen display to become the changes in the file. You change the screen display by moving the displayed cursor to any place in the file and then typing in new text or typing an edit function key. These insertions, deletions and corrections are immediately seen on the screen and become the changes to the file.

You can also perform the common word processing operations of wrapping words at the end of lines and formatting paragraphs between right and left margins. It is easy to print any portion of the text being worked on. Horizontal scrolling allows editing of very long lines. Ten scratchpad buffers may be used for extensive "cut and paste" operations. Powerful search and selective replace operations simplify editing. Other features, such as automatic indenting for structured programming languages, simplify and enhance program development editing.

VEDIT also provides a very flexible and powerful Command Mode, which serves the dual purpose of separating the less commonly used functions from the Visual Mode and of making VEDIT a text oriented programming language. Repetitive editing operations can be performed, blocks of text may be copied or moved within the current file and other files in an almost unlimited manner. The extensive file handling allows multiple files to be edited, split and merged, other files to be viewed and specified portions of other files to be extracted. The command macro capability allows complex editing tasks to be performed automatically. Example of such tasks include numerous search/replace operations on multiple files and source code translations. The command macros can be saved on disk for future use. On line help is available.

You can edit files of virtually any size with little concern over the actual size of the files. You can also recover from common disk write errors, such as running out of disk space, by deleting files or inserting another disk.

Since so many different hardware configurations, keyboards, editing applications and personal preferences exist, VEDIT is supplied with a customization (installation) program in order to let users create versions of VEDIT which are most suitable to their hardware, keyboard, applications and preferences.

CP/M and MP/M are registered trademark of Digital Research, Inc.
MSDOS is a trademark of Microsoft, Inc.

Getting Started

Installation

We supply ready to run versions of VEDIT for some computers such as the IBM Personal Computer, IBM Displaywriter, Apple II, Zenith Z100 and all TRS-80 Models. Otherwise, before you can begin using the editor, you will have to go through the customization process described in Appendix A. Most users with ready to run versions will also want to perform the customization after becoming familiar with VEDIT. While many parameters can be customized, the menu driven operation allows you to limit your attention to a subset of these parameters. In fact, if you are using a CRT terminal and are content to initially use the default keyboard layout (see Appendix F), you only need to select your CRT terminal in order to create a ready to use version of VEDIT. (See "Quick Customization" in Appendix A.) Since the customization process does not destroy or alter the "prototype" editor files on disk, but rather creates a new file with your customized editor in it, you may go through the process as often as you like. As you gain experience with VEDIT you will probably perform the customization several times until you get everything just right.

Using this Manual

This manual is organized into six main sections. The first section describes some basic editing concepts and then introduces the main features of VEDIT and the modes of operation. The second section is a tutorial on the use of VEDIT. The third section covers several moderately technical topics, particularly the text registers and automatic disk buffering. The fourth section describes the Visual Mode in detail, while the fifth section is a detailed description of the Command Mode. The last section contains appendices of the customization process, a reference guide of the commands, a description of the error messages and the index. Your manual may also contain customization notes and keyboard layouts for the many computers VEDIT supports. If so, we recommend that you place those pages which apply to you in a clear plastic protector and file the remaining pages away.

This introductory section also includes a "Sample First Edit Session" to familiarize you with the most basic aspects of using VEDIT. After performing this sample edit session you are best off to at least skim the "Overall Description" in order to get an overview of the capabilities of VEDIT.

Trying out the editor while reading the tutorial section is the best way to gain a working familiarity with most features. The tutorial includes how to invoke and exit VEDIT, and perform the most common editing operations. It also covers some of the file handling,

including splitting and merging files and what to do if you accidentally run out of disk space. The tutorial section is task oriented. Given an editing operation you wish to perform, this section describes which function keys or commands to use to perform the operation. The tutorial should give you enough expertise to use VEDIT very productively.

VEDIT's primary editing mode "Visual Mode" should always be learned first. The Visual (full screen) Mode is easy enough to use that it can be learned by experimenting with the various edit functions, as long as no important files are accidentally altered. The new VEDIT user will have no need for the more complicated "Command Mode", except to exit the editor.

Once you have had some practice with the Visual Mode of VEDIT, you will then want to try out the Command Mode. The Command Mode is definitely not as easy to use as the Visual Mode and more references to this manual will be necessary. However, most basic editing can be done entirely in the Visual Mode, and the Command Mode can be learned gradually as the need arises. Also, the tutorial introduces the most used commands of this mode.

While you will typically spend 99% of your time in the Visual Mode and only 1% in the Command Mode, this manual deals extensively with the Command Mode. This is appropriate, because the Visual Mode is exceptionally easy to learn and use. A little experimentation is the best teacher. The Command Mode, because of its powerful capabilities, is more complex, and more difficult to learn. This manual, therefore, describes this mode in detail with many examples. The large Command Mode section of the manual is intended for the serious command mode user.

The most complex aspect of the Command Mode are the "macros" which can perform repetitive and automated editing operations. Macros are in effect text oriented "programs", which can be developed and saved on disk for later re-use. One type of useful macro is the auto-startup file which can be used to setup programmable function keys on a CRT terminal.

MUCH OF THE COMMAND MODE IS INTENDED AS A "TEXT ORIENTED PROGRAMMING LANGUAGE". YOU SHOULD NOT FEEL COMPELLED TO UNDERSTAND ANY PART OF THIS MODE, NOT DESCRIBED IN THE TUTORIALS, UNTIL YOU ARE QUITE FAMILIAR WITH VEDIT.

We hope that you will enjoy using VEDIT and its many features.

Sample "First" Edit Session

This sample edit session assumes that you have customized VEDIT for your computer system or received a ready to run version. It also assumes that VEDIT is in the file VEDIT.COM (or VEDIT.CMD for CP/M-86) on Drive A: of your computer.

In this edit session you will create a short file with the name of "SAMPLE.TXT", which you can subsequently type out from the operating system.

To create this sample file invoke VEDIT with the command:

```
A>VEDIT SAMPLE.TXT
```

After a short pause in which VEDIT is loaded from disk, it will briefly display the message:

```
NEW FILE
```

This indicates that a new file is being created and not an existing one being edited. VEDIT will then clear the screen and position the "cursor" in the upper left hand corner of the screen. A row of dashes, called the "Status Line" will appear at the bottom of the screen. The status line will display the line and column number for the cursor and the file name, in this case "LINE: 1 - COL: 1" and "SAMPLE.TXT". Having the status line on the bottom screen line indicates that you are in the "Visual Mode" of VEDIT, in which you can perform full screen editing.

You can now begin to type some text which will appear on the screen and will eventually be made into a file. Type in the following text by typing the RETURN key at the end of each line.

> Bach simply awed the professional musicians
> who met or just observed him. Their descriptions
> indicate that Bach, at the head of an orchestra,
> was a conductor very much like the great
> conductors of today.

Altering your text

While typing in the text you can make corrections by using the BACKSPACE key which will move the cursor left and delete the character there. The most important aspect of VEDIT is, of course, that you can easily edit text after it has been typed in. The first step in editing is to position the cursor at the text which needs changing. Although VEDIT has many cursor movement keys, you can get by for now using just the four basic movements UP, DOWN, RIGHT and LEFT. Check your keyboard layout sheet to see what control codes or keys are used

for these four cursor movements. They may already be assigned to the cursor keys on your keyboard, or may be CTRL-E, CTRL-C, CTRL-F and CTRL-S respectively. As you use these keys you will notice that you can only position the cursor at real characters in the text and at the ends of lines. You cannot position it on the screen where there is no text.

With the cursor positioned correctly, you are ready to make your changes. The three basic types of changes are over-writing existing text, deleting text and inserting text. To over-write any text, simply position the cursor at the first character to be over-written and type in the new text. This is the easiest way to correct mis-typed letters. "^" indicates the cursor in the following example:

```
Bach simply wade the professional musicians
           ^
```

After typing the word "awed" the new line will appear as:

```
Bach simply awed the professional musicians
           ^
```

Often you need to insert new text into the middle of a line. For this you use the "Insert Mode" of VEDIT. Look over your keyboard layout and find the key which performs the function [SWITCH INSERT MODE] or [SET INSERT MODE]. Press the key (or control sequence) and the message "INSERT" should appear on the status line. In Insert Mode, text typed in the middle of a line will not over-type any existing text but will shift the text right in order to make room for the new text. Consider the following line:

```
Bach simply awed the professional musicians
           ^
```

After typing the word " all" in Insert Mode, the line will appear as:

```
Bach simply awed all the professional musicians
               ^
```

Deletions are also an important part of editing. VEDIT's two most used deletion functions are [BACKSPACE] and [DELETE]. (Check how your keyboard layout has assigned these functions, [BACKSPACE] is not the BACKSPACE key on Televideo terminals.) [BACKSPACE] deletes the character to the left of the cursor, i.e. the character you may have just typed in. [DELETE] deletes the character at the cursor position. The line always closes up after any type of deletion. Consider the following line:

```
Bach simply aweded the professional musicians
               ^
```

After typing the [DELETE] function twice, the line will be:

```
Bach simply awed the professional musicians
                    ^
```

Although VEDIT can automatically reformat paragraphs, it is good practice to try it manually. Position the cursor in the first line as follows:

```
Bach simply awed the professional musicians
                    ^
```

Type the RETURN key and the line will be split into two lines:

```
Bach simply awed the
professional musicians
^
```

Now position the cursor at the end of the first line and type the [DELETE] function. This will append the second line to the end of the first, giving you your original line back.

Continue making edit changes until the text is modified to your satisfaction.

Saving your text

At this point you are ready to save your text on disk and return to the operating system. For this you must exit Visual Mode and enter the "Command Mode". This is done by the [VISUAL ESCAPE] function which usually corresponds to typing the ESC key twice. Try it. The screen should scroll up (or it may clear) and the cursor will be on the bottom line following the Command Mode "COMMAND:" prompt. At the end of each command line you should type the RETURN key.

One common command is to go back into Visual Mode. The command is: (Remember the RETURN key)

```
V
```

This will put you back into Visual Mode with the status line on the bottom line. The cursor will be positioned at the same place in the text, although not necessarily on the screen, as it was when you exited Visual Mode before. Go back to Command Mode. If necessary, repeat switching between Command and Visual Mode until you can clearly identify which mode you are in.

If you have a printer connected to your computer you may want to print the text you have just created. This can be done from either the Visual Mode or the Command Mode. To print the entire text from Command Mode give the command:

```
B #E0
```

Before giving the command, make sure that your printer is properly connected and "ON LINE".

Finally, while in Command Mode you can give the command to save your text on disk and leave VEDIT. The command is:

EX

You should now be back in the operating system with its "A>" prompt. You can check that your file is on disk with the directory command "DIR", and can type it out with the "TYPE" command, i.e.:

TYPE SAMPLE.TXT

You should see whatever you last saw on the screen in the Visual Mode of VEDIT.

This has introduced you to only a small part of VEDIT's capabilities. However you have experimented with the basics which make up 90% of any editing task, and have tried one of the ways of invoking VEDIT and saving the text on disk. The next section gives an overview of more of VEDIT's capabilities, and the following tutorial gives a hands-on introduction to almost all of the Visual Mode and more of the Command Mode. Good luck.

Notes: A garbled screen display or status line indicates that the CRT terminal is not being supported correctly. Check that you have selected the correct entry in the customization CRT menu. Also be sure that you have selected the correct number of screen lines (use 24 lines for all Televideo terminals).

If after invoking VEDIT you end up in the Command Mode (a "COMMAND:" prompt and no status line), this indicates that you either did not specify a file name after "A>VEDIT", or customized VEDIT to begin in Command Mode (See customization Task 4.3). All of our ready to run versions will start in Visual Mode when a file name is specified.

Overall Description

Introduction

VEDIT is a full screen, or "visual" editor which currently runs under the CP/M, CP/M-86 and MSDOS operating systems and their derivatives, including MP/M, MP/M-86, Concurrent, CDOS and CROMIX. It allows any text file to be created or edited in a visual manner on systems with most types of CRT displays. It has two operating modes: Visual Mode and Command Mode. The typical user will spend 99% of the time in the Visual Mode, the primary editing mode. Here, the screen continuously displays the region of the file being edited, a status line and cursor. Changes are made by first moving the cursor to the text to be changed. You can then overtype, insert any amount of new text and use function keys to perform all changes, which are immediately shown on the screen and become the changes to the file. Several word processing functions such as word wrap and formatting of paragraphs are provided. Ten text registers (scratchpad buffers) allow sections of text to be copied and moved for extensive "cut and paste" type operations. Any portion of the text may be sent to the line printer.

The main purpose of the Command Mode is for performing repetitive editing tasks, explicit file handling and accessing the additional text register operations. The Command Mode allows the execution of common line and character oriented editing commands, including searching, altering, inserting and much more. Single commands and groups of commands may be repeated any desired number of times. A powerful aspect of the Command Mode are command macros, in which simple or very complex command strings are saved in text registers and then selectively executed. Since command macros may be saved and loaded from disk, they can be created and then reused at a later time. The file handling commands allow explicit disk read/write operations, and files to be opened and closed. Another file may be viewed with line numbers and a specified line range of that file can be inserted at any place in the text being edited. Finally, the "EX" command is given to exit VEDIT, saving the edited file on disk.

Basic Editing Concepts

The purpose of editing is to create or modify a file on disk so that it may be saved for future use and processed by another program, such as a print formatter, a compiler, or simply be printed out. When a file is first created, the initial text of the file is entered with the editor, corrections are made, and the text is then saved on disk. When a file is edited, the existing copy of the file is read from the disk into the computer's "main memory", the changes are made with the use of the editor, and the modified text is then saved as a new file on disk. For word processing applications, the text can be printed out before it is saved on disk.

Each file on disk has a name, and when a file is created with the editor, the user assigns the file its name. It is helpful to choose names which are meaningful and easy to remember. The name LETTER1 is thus better than JV%8-G5F. The CP/M and MSDOS operating systems have file names which consist of two parts, the "filename" and the "filetype" or "extension". A "." separates the two parts and the filename may be up to 8 characters long and the extension up to 3 characters long. When a file is to be edited, its name must be specified in order for it to be read from disk. The modified file may be written to disk with a new file name or with the original name. The normal way of invoking and exiting VEDIT will cause it to write it with its original name. One question in this case is: "What happens to the original text file?" VEDIT leaves the original file on disk too, but since you cannot have two files on disk with the same name, the name of the original file is changed to have an extension of ".BAK". This is referred to as the "backup" of the file. Any previous backup of the file on disk will be deleted by this process.

When a file is read from disk, its contents are stored in the "main memory" of the computer. The portion of main memory used for saving the file is referred to as the "text buffer". All changes made to the file are made in the text buffer. When the editing is complete, the file is saved again on disk. This process of reading a file from disk (or creating a new file), making changes to the file and saving it on disk, is referred to as an "edit session". Therefore, two files are being processed while editing. The file being read is called the "input" file and the file being written is called the "output" file. Specifying to the editor which file is to be used for input or output is referred to as "opening" the file. The way VEDIT is normally invoked, i.e. "VEDIT FILE.TXT", the specified file is opened for input, and another file is opened for output which will have the same name as the original input file when the edit session is over. At that time the original input file will still exist, but will have been renamed to a backup file, i.e. "FILE.BAK".

In some cases the file to be edited is too large for all of it to be stored in the text buffer at one time. VEDIT handles such a file by reading the first part of the file into the text buffer, in which you can make any desired changes. After this first part is edited, VEDIT will write the first lines of the text buffer to the output file and read in more unedited lines from the input file. This is repeated until the entire file is edited. If desired, VEDIT can also read edited text back from the output file for further editing. VEDIT can perform this read/ write process automatically and almost invisibly to the user. In particular, when the user reaches the end of the text buffer in Visual Mode, the beginning of the text buffer is written out to disk (to the output file) and more of the file being edited (the input file) is read or "appended" to the end of the text buffer. This process, when done automatically, is referred to as "auto-buffering".

Visual Mode

In Visual Mode, the screen continuously displays the current contents of the file in the region you are editing and a cursor. The bottom line of the screen is used for status information, including the name of the file being edited and the cursor's line number and column position. The changes made to the screen display by typing in new text or using edit functions become the changes to the file. The characters typed while in Visual Mode fall into two categories: Displayable characters and Control sequences. The displayable characters are displayed on the screen at the cursor position and cause the cursor to move to the right. The user customized keyboard layout determines which edit function each control sequence performs. Control sequences are control characters, function keys or escape sequences.

The edit functions fall into two subcategories - cursor movement and visual functions. The cursor movement operations cause no change to the file, but rather move the cursor forward and backward by a character, a word, a line, a paragraph or a screen at a time. Additional cursor movements allow movement to the next tab position, the beginning or end of the text buffer, and to previously set "text markers". Up to ten positions in the text may be marked, so that the cursor can be directly moved to any of these ten positions.

VEDIT has two modes for inserting new text: "Insert" mode and "Normal" mode. In normal mode, the new text will overwrite the existing text. In insert mode, the existing text is not overwritten, but rather is squeezed to the right as the new text is typed. New lines are started by simply typing the RETURN key. Typing the RETURN key in the middle of a line splits it into two lines.

Text can be deleted on a character, word, line or block basis. The character to the left or at the cursor position can be deleted, and the word to the left or right of the cursor can be deleted. An entire line, or only the portion to the right of the cursor can be deleted.

A useful feature is the ability to move or copy a section of text to any other position in the file. ("Copy" implies that the original text is not deleted, while "Move" implies that the original text is deleted.) This is done by first copying, moving or appending the text to one of the ten text registers (scratch pad buffers), and then inserting the text register at any place or places in the file. (It may also be inserted in another file). Blocks of text are deleted by moving the block to a text register and emptying the text register, or just "forgetting" about it. Any portion of the text can easily be printed on the line printer and special printer control characters can be imbedded in the text.

For word processing uses, the Visual Mode can perform word wrap and formatting of paragraphs between adjustable right and left

margins. When word wrap is on, VEDIT will move an entire word which didn't fit within the right margin to a new line, while you are typing. Formatting a paragraph also wraps words, but operates on existing paragraphs and the currently set left and right margins. You can therefore change the margins and re-fit the paragraphs within the new margins.

Search and Replace operations can be performed in Visual Mode. The next or "nth" occurrence of a "string" can be searched. Extensive pattern matching is provided. The Replace operation prompts the user whether each occurrence of the original text is to be replaced with the new text. Replacement can also be made without prompting.

Several options are available for dealing with the Tab character. Normally when the "Tab" key on the keyboard is typed, a tab character is placed into the text. A tab character is displayed as spaces to the next tab position. They may be set as desired; and are normally at every 8th position. Optionally, typing the "Tab" key can have VEDIT insert spaces to the next tab position into the text.

VEDIT has several unique built in aids for program development. One is automatic indentation for use with structured languages such as Pascal, PL/I and C. When "Indenting" is set, the editor will automatically insert tabs and spaces to the current indent position following each RETURN. The indent position can be moved right and left by an adjustable indent increment. Many assembly language programmers prefer their program code to be in upper case letters with comments in upper and lower case. VEDIT can accept all text in lower case and automatically convert the labels, opcodes and operands to upper case while leaving the comments in lower case. It does this by searching on the line being entered or edited for a special character such as ";". To the left of the ";" lower case letters are converted, to the right of the ";" they are not converted. This is referred to as "Conditional Lower to Upper Case Conversion".

The Visual Mode can handle lines which are up to 260 characters (256 plus CR LF and two spare) long. The screen can be scrolled horizontally (or sideways) for editing files with long lines, such as spreadsheets. The screen can be scrolled sideways up to a maximum margin of 255. This "horizontal scroll margin" is changeable.

Text lines longer than the horizontal scroll margin are "wrapped" to the next screen line and are called "continuation lines". VEDIT indicates continuation lines by placing the "-" character, often in reverse video, in the reserved first column of the screen. The screen will automatically scroll sideways or create continuation lines as you enter text.

Note: If you are just beginning to use VEDIT, you may wish to skip to the "Visual Mode Task Tutorial" at this point.

Command Mode

Command Syntax:

In Command Mode, the user enters command lines which consist of single commands, concatenated commands or iteration macros. Each command line, whether it consists of one command or multiple commands ends with a RETURN or <ESC> <ESC>. Commands which specify a "text string" may require an <ESC> following the text string.

Each command consists of a single letter or two letters if the first letter is "E", "R", "X", "Y" (Extended, Register, Numeric and Misc commands). Some commands may be preceded by a number, called the "iteration count", to signify that the command is to be repeated. If no number is given, a "1" is used as the default. Wherever a number is allowed, you can also use the "#" character to represent the maximum positive number 32767. Other commands take the preceding number as a numeric argument. Several commands are followed by additional arguments such as text strings, files names or text register numbers.

Multiple commands may be typed one after another on a command line. They are always executed left to right. Their effect is the same as if each command had been typed on its own command line.

It is also possible to repeatedly execute a group of commands. This is done by enclosing the desired group of commands within brackets "[" and "]". Such a group of commands is called an "Iteration macro". The initial "[" is preceded by a number, called the "iteration number", which specifies how often the group of commands will be repeated.

(Note: The characters for enclosing iteration macros are printed as "[" and "]" in this manual. Some users may be more familiar with angle brackets "<" and ">" and can choose either set during customization.)

It is often desirable to use a sequence of commands, particularly iteration macros, over and over again. This can be done by storing the sequence of command in one of the ten text registers and then executing the commands in the text register. Any sequence of commands executed in a text register is called a "Command macro". Command macros may be stored on disk and loaded back into a text register for later re-use. Command macros may be thought of as "programs". This command macro capability in fact makes VEDIT a very powerful text oriented programming language.

Command Operation:

Many of the commands make a change to the text buffer at the position determined by the "edit pointer". The edit pointer is very much like the cursor in Visual Mode, it is just not as readily seen. Commands exist to move the edit pointer a character at a time, a line at a time or to the beginning or the end of the text buffer. The number of lines or characters the edit pointer moves is determined by the iteration number for the command. Negative iteration numbers mean backward movement, towards the beginning of the text buffer. One command types out a given number of lines before or after the edit pointer to display the contents of the file and "show" the user where the edit pointer is.

The commands which alter the text all operate from the position of the edit pointer. One deletes characters, one deletes lines, one inserts new text and another searches for a string of characters and changes them to another. Other commands only perform searching without alteration. Ten commands are available for dealing with the text registers. Three commands are used to change the various switches, parameters and tab positions which VEDIT uses in both Command and Visual Modes. One command puts the editor into Visual Mode. The last two groups of commands deal with the reading and writing of files and with the opening and closing of input and output files.

The commands fall into ten overlapping categories:

Edit pointer movement	-	B, L, C, Z
Display and Print text	-	T, EO
Alter text	-	D, I, K, S, EI, YI
Search / Replace	-	F, N, S
Text Register	-	G, M, P, RD, RI, RL, RP, RS, RT, RU, R*
Disk Buffering	-	A, N, W, EA, EQ, EX, EY, EZ
File Handling	-	EB, EC, ED, EF, EG, EK, EL, ER, EW
Switch and Tab Set	-	EP, ES, ET
Numeric Registers	-	XA, XS, XT
Misc	-	YI, YL, YR, YS, YT, YW, U, V

The "V" command enters the Visual Mode, and the "U" command prints three memory usage numbers.

Which Mode to Use for What

The Visual Mode is designed to satisfy the majority of all editing needs. The bulk of editing consists of inserting new text, correcting typos, and making revisions, which includes moving blocks of text around. These are all readily handled in Visual Mode and are best done in that mode. There is probably a three to one time savings in inserting new text and correcting the typos in Visual Mode over Command Mode. There is probably a ten to one time savings in making the revisions in Visual Mode, compared to Command Mode, even assuming you are very practiced with the commands!

Common search and replace operations are also best done in Visual Mode. Searching is used to directly access a particular word or string in the file. Replace operations are more flexible in Visual Mode since the replacement may be selectively done, where the user is prompted whether each occurrence of the "old" string should be replace with the "new string."

Any edit operation which can be performed in Visual Mode can also be performed in Command Mode. However, straight forward modifications, insertions and deletions are much easier done in the Visual Mode. Unless they are part of iteration macros or command macros, the equivalent of the "L", "C", "T", "D", "I", "F" and "S" commands are best done in Visual Mode.

Command Mode is most useful for performing repetitive edit changes using macros and for extensive file handling. Command Mode is also used to change the various VEDIT switches, parameters and tab positions. Text register operations such as loading and saving them on disk can only be performed in the Command Mode. The edit pointer in Command Mode and the cursor in Visual Mode both serve a similar purpose. When entering Visual Mode, the cursor takes on the position in the text buffer of the edit pointer in Command Mode. When exiting Visual Mode to Command Mode, the edit pointer takes on the last position of the cursor.

Command Mode is also used when the editing process involves more than just making changes to a single file. The file handling commands allow several files to be merged into one file or a file to be split into several smaller ones. Portions of one file can be copied to a text register and the text register subsequently can be inserted into another file. Another file may also be viewed with assigned line numbers. A specified line range of that file may then be directly inserted into any place in the file being edited. Many other possibilities exist and some examples are given in the "Detailed Command Description" of this manual.

Word Processing with VEDIT

VEDIT can be used for two types of word processing. One is stand alone word processing in which the text is composed entirely with VEDIT and then printed out exactly as it appears on the screen. Alternately, the supplied print formatter macro PRINT.EXC can be used to automatically handle page headings, page splits and page numbers. Other than these page breaks, the text has to be formatted exactly the way it is to be printed out. This includes the centering of lines, and other details which VEDIT does not perform automatically. VEDIT can, however, format paragraphs between left and right margins. Therefore, if a paragraph is currently ragged, with very different line lengths, VEDIT can format the paragraph between any left and right margins. If after formatting, you decide, for example, that you now want the paragraph indented on both sides, VEDIT can also do this for you automatically. The optional mail merge macro VMAIL can be used for creating and printing form letters.

The second type of word processing uses a "Print Formatter" such as CompuView's V-PRINT, which takes care of the details of page layout, such as page headings, centering of lines, justification and more. In this case VEDIT is used to create a file which contains the text and short command lines to the Print Formatter, which does the final printing. VEDIT's facility for word wrap still makes it easier to enter the text, but the formatting of paragraphs becomes mostly cosmetic since the Print Formatter generally formats its own paragraphs. VPRINT takes commands which begin with a period "." in the first column of a line. When formatting paragraphs, VEDIT recognizes such commands lines and leaves them alone.

VPRINT not only handles the details of page layout, but also has facilities for automatic index and table of contents generation, underlining, multiple font support for popular printers and much more. Combining VEDIT with VPRINT will give you more word processing capabilities than found on nearly any stand alone word processors. As text documents become longer, most stand alone word processors become increasingly inefficient, because they can only edit a very small portion of the text at one time. Some even have an upper limit to the size of text they can handle. With longer documents, such as manuscripts, using VEDIT with VPRINT will get the job done faster and with less effort. CompuView's V-SPELL spelling checker/corrector lets you quickly correct the spelling in any document. It displays the error words in context and gives suggested spellings. Since it also corrects the document, there is no need to go back to VEDIT following the use of V-SPELL.

V I S U A L A N D C O M M A N D M O D E T A S K T U T O R I A L

Vedit Tutorial

This section is a tutorial on the basic editing capabilities of VEDIT. It is task oriented and gives the commands necessary to perform simple editing operations such as inserting text, and more complex tasks such as moving text and concatenating files. As a "Hands-On" tutorial, it is meant to be followed while actually running VEDIT. Later, as a reference, it explains how to combine commands in order to perform a desired task.

Not every possibly conceivable text editing situation or sequence of commands is included here. However, we have tried to include a comprehensive list of editing tasks --- some elementary, others with many steps. Tasks are presented so that you should rarely have to look forward in this section to learn something necessary for the completion of the current task. For example, moving the cursor is the first task discussed; it is used in almost every following task.

The labeled boxes in this section represent visual edit functions, such as [CURSOR UP] and [INDENT]. The actual keys you type to perform the functions are chosen in the VEDIT customization procedure. If you are using one of our example keyboard layouts, refer to the layout sheet. Most layouts use both control characters and escape sequences. Control characters such as <CTRL-Q> are typed by holding down the CONTROL key while typing the "Q". Escape sequences such as ESC-R are typed by first pressing the "ESC" key and then the "R".

The "ESC" key is also used in some command mode commands to mark the end of text "strings". It is represented in all examples in this manual as an "$", which is also what VEDIT displays on the screen when an ESC is typed in command mode.

Invoking VEDIT

To use VEDIT it has to be invoked from CP/M or MSDOS with the proper command. The next page describes all the ways of invoking VEDIT, but the most common is just to type "VEDIT" followed by the name of the file to be edited or created. For example: (The "A>" is the prompt given by PCDOS or CP/M).

```
A>VEDIT LETTER.TXT
```

VEDIT will then read in the file "LETTER.TXT", or if you are creating the file, briefly display the message "NEW FILE". It will then normally go into the "Visual Mode" which displays the beginning of the file on the screen. The bottom line will contain the "Status Line" which consists mostly of dashes "-", and optionally the line and column numbers. Unless you have a short 40 column screen, the file name "LETTER.TXT" will also appear on the status line. Also visible will be the "Cursor" which indicates at what position on the screen you are editing. It will initially be in the upper left hand corner. At this point you are ready to begin editing.

For the purposes of this tutorial it will be best if you begin by editing a file which already exists, instead of creating a new one. If you don't have any such files available, you can copy one of the files with a file name extension of ".DOC" from your VEDIT distribution disk to your work disk. Don't be concerned about making accidental changes to the file, because you can easily quit the edit session in such a way that no files are actually changed. (We assume, of course, that you have made a copy of your distribution disk.)

INVOKING VEDIT

VEDIT FILENAME.EXT	You will land in Visual Mode (status line will appear at the top or bottom of screen) OR Command Mode ("COMMAND" prompt), depending on the parameter set by command ES. See "Command Mode Detailed Command Description".
VEDIT	Begin in Command Mode. Choose a file to edit with an "EB filename" or perform any other Command Mode command.
VEDIT INFILE.EXT OUTFILE.EXT	"INFILE.EXT" will be read in and not altered, while "OUTFILE.EXT" will be created. If "OUTFILE.EXT" already exists, it will be renamed to "OUTFILE.BAK". This form is equivalent to invoking VEDIT without any filenames (second form) and then issuing the command: ER infile.ext$ EW outfile.ext Use this form if the edited file is more than half a disk long. In this case, INFILE.EXT is the file to be edited and OUTFILE.EXT is specified to be on another disk drive with a nearly blank disk.

Keyboard Characters

In Visual Mode, you edit the file by performing two basic types of operations: entering new text, or performing edit functions by typing control sequences. All the letters, numbers and other normal characters on your keyboard can be directly entered as new text. Go ahead and try typing a few words in right now. Notice that as each character is typed, it appears at the cursor position and the cursor then moves to the right. If there already were characters on the line, you have just overwritten them. We will soon see that it is just as easy to insert characters without overwriting. The control sequences are used to perform the various editing functions. The keyboard layout that you have customized determines which editing function each control sequence performs. Control sequences can be control characters, such as <CTRL-S>, escape sequences such as ESC-P, or a function key on your keyboard. Function keys generally send a control character or an escape sequence when you type them.

Editing Functions

The editing functions in the visual mode break down into two categories. One type are the "Cursor movement" functions which only move the cursor around on the screen and scroll the screen to display different parts of the file, but do not change the file in any way. Look at the keyboard layout you are using and try typing the control codes for some of the cursor movement functions such as [UP] [DOWN] [RIGHT] and [LEFT]. The following pages describe all of the cursor movements, and you are advised to briefly try them all out. Don't be concerned about remembering them all now. Some are more important than others, and you will get along quite well knowing only [UP], [DOWN], [RIGHT], [LEFT], [ZIP], [PAGE UP] and [PAGE DOWN].

CURSOR MOVEMENT

Operation	Command Sequence
Move cursor right	CURSOR RIGHT
Move cursor left	CURSOR LEFT
Move cursor up	CURSOR UP
Move cursor down	CURSOR DOWN
First character of current line	BACK TAB
Move cursor to next tab position	TAB CURSOR
Last character of current line	ZIP
First and last character of current line	LINE TOGGLE
First character of next line	NEXT LINE

Operation	Command Sequence
First character of the previous word	PREV WORD
First character of the next word	NEXT WORD
Beginning of current paragraph	PREV PARA
Beginning of next paragraph	NEXT PARA
Top and bottom screen lines	SCREEN TOGGLE
Move cursor up by scrolling	SCROLL UP
Move cursor down by scrolling	SCROLL DOWN
Move cursor right by scrolling horizontally	SCROLL RIGHT
Move cursor left by scrolling horizontally	SCROLL LEFT

PAGE MOVEMENT

Purpose: To rapidly access other regions of the file not currently displayed on the screen.

Operation	Command Sequence
Previous Page of text	[PAGE UP]
Next Page of text	[PAGE DOWN]
First Page of text (First character)	[HOME]
Last Page of text (Last character)	[ZEND]
Set invisible text marker at cursor position	1.) [SET TEXT MARKER] 2.) Type digit "0 - 9" to specify which marker to set. Or set marker "0" by typing [SET TEXT MARKER] or RETURN.
Move cursor to previously set text marker	1.) [GOTO TEXT MARKER] 2.) Type digit "0 - 9" to specify which marker to goto. Or goto marker "0" by typing [SET TEXT MARKER] or RETURN.

Entering New Text

The three edit functions relating to the "Insert Mode" give you two choices for switching between the "Insert" and "Normal modes". You started in Normal mode, and the displayable characters you typed over-wrote any existing characters. When you switch to Insert mode you will see the word "INSERT" on the status line and any character at the cursor position will be squeezed to the right when you type in new characters. Try it to see the difference between the two modes.

You may be wondering about how to insert entire lines into the text. To start a new line you simply type the RETURN key. If the cursor is at the end of a line, this opens up a blank line on the screen on which you can enter text. If you enter a lot of new lines, one after another, the screen will automatically scroll to keep up with you. If the cursor is in the middle of a line when you type RETURN, the line is split into two lines, with the character at the cursor position and all following characters moving to the new line. With the [DELETE] function, explained in a few pages, you can also concatenate lines together.

ENTERING NEW TEXT

Operation	Command Sequence
Entering text into the text buffer --- beginning an empty file or continuing at the end of a file.	NONE - Move cursor wherever you like and begin typing. What you see is what you get.
Overtyping (typing over existing text)	1.) Position cursor over first character to be overtyped. 2.) Retype.
Inserting new characters in between existing characters	1.) INSERT — Watch for "Insert" prompt on status line 2.) Type new text 3.) INSERT — "Insert" prompt disappears (or leave INSERT on.)

Visual Functions

The second category of editing functions are called the "Visual Functions" which perform such operations as deleting characters or lines, indenting on the left side and moving sections of text to other parts of the file. The following pages describe each of these functions.

Deleting Text

VEDIT has functions to delete the character at the cursor position, the previous character, the previous word and the next word. Two functions will delete partial or entire lines. These are described on the next page. Go ahead and try out the [DELETE], [BACKSPACE], [EREOL], and [ERLINE] functions. Notice that the [UNDO] function will bring back the original text on the line unless you erased the entire line with the [ERLINE].

You can delete an entire line with the [ERLINE] function. You can also concatenate two lines by moving the cursor to the end of the first line and typing [DELETE]. Go ahead and try all of this, especially splitting lines with a RETURN and concatenating lines with a [DELETE].

Paragraphs or blocks of text are deleted by moving them to a text register and then emptying the text register.

DELETING TEXT

Operation	Command Sequence
Delete character at left of cursor; shift following characters left	BACK SPACE
Delete character at cursor; shift following characters left	DELETE
Erase from cursor to end of line	EREOL
Erase entire line cursor is on and close up text	ERLINE
Delete word to left of cursor	DEL PREVIOUS WORD
Delete word to right of cursor	DEL NEXT WORD
Delete paragraphs and blocks of text	1.) Position cursor over first character in the paragraph to be deleted. 2.) MOVE TO TEXT REGISTER

3.) Position cursor past last character in paragraph to be deleted.

4.) [MOVE TO TEXT REGISTER]

5.) Type digit "0 - 9" to specify which text register to use. Or use reg. "0" by typing [MOVE ...] or RETURN.

6.) [MOVE TO TEXT REGISTER]

7.) [MOVE TO TEXT REGISTER]

8.) Type same digit "0 - 9", or [MOVE ...] or RETURN to empty the text register.

CORRECTING MISTAKES MADE TO A LINE

[UNDO]

This command returns the line the cursor is on to its appearance before the cursor was most recently moved to that line.

This does not mean that, by putting the cursor on a previous line you changed, [UNDO] will give you the original line.

REPEATING OPERATIONS

Purpose: It is often necessary to repeat an edit operation such as inserting the same character, deleting many lines, or moving the cursor many "pages" forwards or backwards. By using the [REPEAT] function key, you can perform these repeated operations without having to type the same key over and over again. Pressing the [REPEAT] key once gives a repeat value of "4" (see status line). Pressing it again multiplies the value to "16", then to "64" and finally "256". Any other value may be selected by typing a number between 00 and 256. Once the repeat value is correct, simply type the desired key or control sequence which is to be repeated.

Operation	Command Sequence
Delete four lines of text	[REPEAT] [ERASE LINE]
Page forward by 16 pages	[REPEAT] [REPEAT] [PAGE DOWN]
Delete 30 characters	[REPEAT] [3] [0] [DELETE]
Insert 20 blank lines	[REPEAT] [2] [0] [RETURN]
Insert 40 "*" characters	[REPEAT] [4] [0] [*]
Re-format the next 10 paragraphs	[REPEAT] [1] [0] [FORMAT PARAGRAPH]

Note: If you want to cancel the [REPEAT] operation type in 000 and any character, or press [CANCEL].

Indenting Text

If you don't want your text to begin in the first column, you can let VEDIT automatically indent your text with the [INDENT] and [UNDENT] functions. The section "Visual Mode - Indent and Undent" explains these functions, but it is easier to understand them through experimentation. Type a RETURN to start a new blank line, then type the control sequence for [INDENT]. Notice that the cursor has moved right by 4 spaces to column 5 (unless you have changed this parameter). Type a few words and another RETURN. This time the cursor will begin immediately in column 5. You have set the "Indent Position" to column 5, and it will stay there until you increase it with another [INDENT] or move it back with [UNDENT]. To achieve the indentation, VEDIT inserts the most Tabs and fewest spaces to the indent position. You can confirm this by moving the cursor over these leading Tabs and spaces, and if you like, you can also delete them or insert characters before this "Left margin". VEDIT only creates this indentation when you type RETURN, when "Word Wrap" is being used and when paragraphs are formatted.

INDENTING TEXT BLOCKS

Operation	Command Sequence
Increase the amount of Indentation. (Move left margin to the right)	INDENT
Decrease the amount of Indentation. (Move left margin to the left)	UNDENT
To change Indent/Undent increment:	
1.) Enter command mode	VISUAL EXIT
2.) Issue command, where n= # of columns indented each time. EX: EP 3 4 will indent to 5th column, 9th column, etc.	EP 3 n$$
3.) Enter visual mode again	V

WORD WRAP AND FORMATTING PARAGRAPHS

Note: Before the Word Wrap or the Format Paragraph function will work, the right margin must be set in command mode (unless it was set during customization). The left margin is set with the INDENT and UNDENT functions. A right margin of 00 turns Word Wrap off and disables the Format Paragraph function.

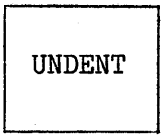
1.) Set right margin and invoke word wrap at column n. This allows n columns to be used. Word wrap begins at column n + 1.

EP 7 n

If "n" = 00 word wrap is disabled.

2.) Move left margin to the right.

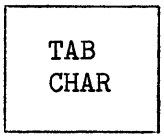
3.) Move left margin to the left.



4.) Set addition indent for the begin of a paragraph with spaces.

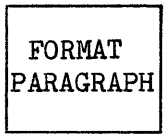
-- OR --

Use a tab. See ET command to change tab positions.



5.) Type in the paragraph. Words will be wrapped as needed.

6.) Reformat a paragraph to current left and right margins.



Moving and Copying Blocks of Text

A useful facility in VEDIT is the ability to move blocks of text to other regions in the file, to duplicate blocks of text and to delete blocks of text. These are done through the use of the "Text Registers" and the functions [COPY TO TEXT REGISTER], [MOVE TO TEXT REGISTER] and [INSERT TEXT REGISTER]. The text registers are simply regions in memory in which VEDIT can store text which is independent of the main text you are editing. A block of text is any amount of text from one character to an entire file. You can COPY a block of text to a text register, in which case your main text is unaltered, or you can MOVE a block of text to a text register, in which case it is also deleted from your main text. Alternately, these copy and move operations can append the block of text to any existing text in the register. At any time you can insert a text register into your main text, which does not alter the text register.

The following page describes the steps to copy a block of text from one area of the file to another. Note that at step (3.), the cursor must be positioned just AFTER the block of text. If you wish to include the end of a line, this would be the first column of the following line. You could of course position the cursor at the end of the line, but in this case the carriage return which ends the line would not be included in the text move.

If you wanted to move the text, you would use the same procedure, except use the [MOVE TO TEXT REGISTER] function instead of [COPY TO TEXT REGISTER]. In this case the text will also be deleted from your main text and from the screen.

If you type [COPY TO TEXT REGISTER] or [MOVE TO TEXT REGISTER] twice at the same location, the specified text register will be emptied. If all registers are empty, the "TEXT" message will disappear from the status line. You can therefore delete a block of text by moving it to a text register and then emptying the text register.

Since there are ten text registers, the status line will prompt you for a digit "0 - 9" to specify which text register to use. If you only need to use one register, you can simply answer the prompt by typing the function key again or a RETURN to specify register "0".

In practice it doesn't matter whether you type [COPY TO TEXT REGISTER], [MOVE TO TEXT REGISTER] or even [PRINT] at step 2. Only after you have marked one end, and the message "1 END" is on the status line, must you be sure to type the correct key. If you do type the incorrect key and get the status line prompt, you have two options. You can type the DEL key on your keyboard (if you have one - not necessarily the same as the [DELETE] function), the prompt will clear and you get another chance to type the correct key. Otherwise type [CANCEL] to cancel the entire function.

MOVING TEXT WITHIN THE FILE

1.) Position cursor over first character in block to be moved.

2.) | MOVE TO TEXT REGISTER | — Message "1 END" on status line

3.) Position cursor past last character in block to be moved.

4.) | MOVE TO TEXT REGISTER | — Message "REGISTER [+] 0-9" on status line

5.) Type a digit "0 - 9" to specify which register to put text into, or [MOVE ...] or RETURN to use register "0". — Type optional "+" before digit if new text is to be appended to any existing text in register, instead of overwriting it.

6.) Position cursor at position to insert the text.

7.) | INSERT TEXT REGISTER | — Message "REGISTER [+] 0-9" appears

8.) Type the digit of the register you wish to insert in the text.

See NOTES on next page.

NOTES:

1.) If you get a "FULL" message at step 4, there is insufficient memory for the Text Register to contain the entire text block. Nothing was inserted into the Text Register. See task "Making More Memory Space".

2.) Following the text insert in step 6, the cursor is positioned at either the beginning or end of the inserted text depending upon ES command switch 4.

3.) In step 3, in order to include the CR-LF of the line, position the cursor at the beginning of the next line.

4.) Alternately you may reverse steps 1) and 3), i.e. either end of the block may be set first.

5.) [CANCEL] will cancel the operation any time "1 END" appears or you have a status line prompt.

EMPTYING A TEXT REGISTER

Purpose: It is best to empty a text register when its contents are no longer needed. This frees up more memory space too.

1.) | MOVE TO TEXT REGISTER |

2.) | MOVE TO TEXT REGISTER |

Type command key twice with cursor at same position to empty the register.

3.) Type a digit "0 - 9" to specify which register to empty out. Or type [MOVE ...] or RETURN to empty register "0".

SENDING TEXT TO THE PRINTER

1.) Be certain your printer is on, and the "on line" or "select" function on the printer is enabled. (See your printer manual).

2.) Position cursor at beginning of text block.

3.) | PRINT TEXT | — Will get "1 End" message on status line.

4.) Position cursor at end of text block.

5.) | PRINT TEXT | — Printer should start now.

6.) <CTRL-C> — To stop the printing.

ENTERING CONTROL CHARACTERS INTO THE TEXT

1.) | NEXT CHAR LITERAL | — This will enter the next character into the text, even a control character. You probably want to be in INSERT mode before this command.

2.) Type a control character which will be entered into the text at the cursor position.

SEARCHING

The search or replace operation always moves forward from the current cursor position.

1.)	FIND	Status line will clear and give FIND? prompt.
2.)	Type up to 30 characters that you are searching for. Pattern matching codes are allowed. End with a RETURN. Use <CTRL-N> to search for RETURN.	Screen will rewrite with cursor past the found text. Will give error if text not found - You must then type any key to continue.
3.)	FIND	Type [FIND] to search for the next occurrence. To find "nth" occurrence use [REPEAT] n [FIND]
4.)	CANCEL	Allows another string to be searched. Automatically cancels if string not found.

If you enter an immediate RETURN in Step 2, you can select from the search options. The prompt changes to:

OPTIONS (Begin / Global / Reuse)?

Select one or more options by typing the corresponding letters "B", "G", and "R", followed by a RETURN.

Begin Allows the search to start at the beginning of the text buffer. If the "Begin" is preceded by the "Global" option, the search starts at the beginning of the file.

Global Allows the search to operate to the end of the file, if necessary, instead of just the end of the text buffer.

Reuse Allows the previous searched characters to be reused. Useful following a "CANNOT FIND" message to restart the search from the beginning of the file with the "Begin" option.

Following the options and a RETURN, the prompt will change back to "FIND?", unless the "Reuse" options was selected, in which case the previously entered characters will be searched for again.

REPLACING

1.) REPLACE

Status line will clear and give FIND? prompt.

2.) Type up to 30 characters that you are searching for. Pattern matching codes are allowed. End with a RETURN. Use <CTRL-N> to search for RETURN.

Screen will rewrite with cursor past the found text. Will give error if text not found - You must then type any key to continue.

Status line will now prompt "REPLACE WITH? "

3.) Type up to 30 characters which is the replacement string. End with a RETURN.

Screen will rewrite as in step 2.)
Screen now prompts:
REPLACE (Y / N / REST / CANCEL)

4.) Type "Y" to make the replacement, "N" not to make it, "R" to replace this and all following occurrences, "C" to cancel the operation.

5.) REPLACE

Type [REPLACE] to search for and prompt for the next replacement. To prompt for "nnn" replacements use:
[REPEAT] nnn [REPLACE]
in step 1.

6.) CANCEL

Allows another string to be replaced. Automatically cancels if string not found.

ENTERING COMMAND MODE

Besides the "Visual Mode" in which all editing is done on the screen at the cursor position, VEDIT has a command mode, where all editing is done by typing in command lines which end in RETURN or <ESC> <ESC>. The command mode is definitely not as easy to use as the visual mode, but fortunately you don't need to know very much of it in order to use VEDIT very successfully. One thing you do have to know is how to enter command mode in order to end the edit session. This is done by typing the control code for [VISUAL EXIT] or [VISUAL ESCAPE]. Go ahead and try it. The screen will scroll up one line and the command mode prompt "COMMAND:" will appear below the status line. The command to go back to visual mode is "V" (remember the RETURN). Notice that the cursor is at the same position in the text (but not necessarily on the screen) as it was when you exited visual mode.

SWITCHING FROM VISUAL MODE TO COMMAND MODE

Exit visual mode into command mode. "Edit Pointer" takes on last position of cursor.	
Same as above, but also aborts any command, such as an iteration or a macro.	

SWITCHING FROM COMMAND MODE TO VISUAL MODE

Command to enter visual mode. Text registers are preserved. Cursor takes on last position of command mode "Edit pointer".	V

SAVE ALREADY EDITED TEXT AND CONTINUE

Purpose: You should make it a habit to regularly save your text on disk during a long edit session. This way you will lose less work in case of a power or hardware failure, or if someone accidentally turns off the computer. Saving the text every hour and whenever you leave the computer is suggested.

1.) | VISUAL
EXIT | — If in visual mode, enter command mode.

2.) EA — Write file to disk; same file will be used to continue edit session.

BEGIN EDITING NEW FILE

Purpose: It is not necessary to exit VEDIT and invoke VEDIT again from the operating system in order to edit another file. Since the contents of the text registers are not lost when you begin editing another file from within VEDIT, it is very easy to copy or move portions of one file to another.

1.) | VISUAL
EXIT | — If in visual mode, enter command mode.

2.) EY — Save current file on disk and empty the text buffer.

3.) EB newfile.ext — Begin editing the file "newfile.ext", which may be an existing file, or a file to be created. Note: a space before the filename is allowed.

4.) V — Enter visual mode for full screen editing of the file.

MAKING MORE MEMORY SPACE

Purpose: When using the text registers extensively, you may run out of memory space for performing the desired operations. This is usually indicated by a *BREAK* in command mode, or a "FULL" in visual mode. First try and empty any text registers which are no longer needed. If this does not give you enough memory space, you can write out the first part of the text if it is already edited.

1.) Position cursor past end of text which does not need changing (it's been corrected already).

2). | VISUAL EXIT | — Enter command mode.

3). OW — Write this text out to disk. More room is now available.

4.) V — Enter visual mode for full screen editing of the file.

DISK DIRECTORY DISPLAY

Purpose: The disk directory on any drive can be displayed. This is useful when editing, merging or splitting multiple files. In the event that you run out of disk space, you can then also see if any files can be deleted.

Operation	Command Sequence
Display directory on default drive.	ED
Display directory on drive B:	ED B:
Display directory of all files with extension ".ASM" on drive A:.	ED A:*.ASM

INSERT A LINE RANGE OF FILE 1 INTO FILE 2

Purpose: It is often desirable to insert a portion of another file, such as a paragraph or a subroutine, into the text being edited. VEDIT lets you extract a specified line range of another file and insert it into your text. The "EL" command to used to look at another file with line numbers, and the "EG" command is then used to insert the desired line range.

Note: "file1" can include a drive specifier and CP/M user number.

1.) EL file1[m,n]	The line range 'm' through 'n' of 'file1' is typed on the screen. If '[m,n]' is not specified, the entire file is typed out. Specifying a line range lets you zero in on the correct lines without displaying the entire file. Use <CTRL-S> to stop/start the screen display. Note the desired line range.
2.) EG file1[m,n]	Lines 'm' through 'n' of 'file1' will be inserted into the text at the edit (cursor) position. To copy an entire file, leave off the '[m,n]'. Note: the comma in '[m,n]' can be replaced by a space, i.e. '[m n]'.

If you get a *BREAK* message there was insufficient memory space to insert the entire text, and as much as possible was inserted. To make more space for other files, text, etc., try emptying some of the text registers or writing the first part of the text out to disk, as described earlier.

CONCATENATING TWO FILES

Purpose: It is sometimes desirable to append one file to the end of another. This is readily done with VEDIT. In this example the text in file "file2" is appended to the end of the text in "file1" and the combined text is written to the file "file3". The three files can be on different disks.

Note: This assumes that the entire file 'file1' fits into memory.

1.) VEDIT	Invoke VEDIT without a filename. VEDIT will come up in command mode.
2.) ER file1$0A	Setup the first input file for reading, and read it in. This assumes that the entire 'file1' fits into memory.
3.) (optional) EC	Only needs to be done if the disk with 'file2' is not in one of the drives. After the EC, make sure that the disk with 'file2' and the disk to hold 'file3' are in the drives.
4.) EW file3	Setup the output file which will hold the combined text.
5.) ER file2$0A	Read the second input file. All of it does not need to fit into memory.
6.) EX	This writes out the complete file 'file3' and exits VEDIT.

Note: In the example above, "$" is the <ESC> key, and "0A" is the digit zero followed by "A".

SPLITTING A FILE INTO TWO OR MORE FILES

Purpose: VEDIT allows you to split a large file into several smaller ones. This example assumes that the splits are simple -- the front, middle and end sections of a large file are copied to their own files. More complex splitting can be done with the text registers. In this example 'file1' is split into 'file2', 'file3' and file4'.

1.) VEDIT	Invoke VEDIT without a filename. VEDIT will come up in command mode.
2.) ER file1$OA	Setup the large input file for reading, and read it in. The entire 'file1' need not fit into memory.
3.) EW file2	Setup first output file.
4.) V	In visual mode, position the cursor on the first character of the second part of the large file. Return to command mode.
5.) OW EF OA EW file3	Write the first part of 'file1' to 'file2' and close it. OA will read read in more of 'file1' if necessary. Setup the second output file.
6.) V	See step 4. Not needed if only splitting into two parts.
7.) (optional) OW EF OA EW file4	Not needed if splitting into two parts. Write the 2nd part of 'file1' to 'file3' and close it. OA will read in more of 'file1' if necessary. Setup the third output file. (Repeat as necessary.)
8.) EX	Write the rest of 'file1' to the last output file and exit VEDIT.

Note: In the example above, "$" is the <ESC> key, "OW" and "OA" are the digit zero followed by "W" or "A".

RECOVERY FROM FULL DISK ERRORS

Purpose: If you attempt to write more text to disk than the disk can hold, you will get a "NO DISK SPACE" error and a return to command mode. Occasionally you may get a "NO DIR SPACE" error which means the disk has insufficient directory space to hold the rest of the file. You can recover from both of these errors by deleting old files on the disk, or by writing the rest of the file to another disk.

1.) EX NO DISK SPACE *BREAK*	You attempt to finish the edit session, but you get the disk full error.
2.) ED	Issue the directory command to see what files can be deleted.
3.) EK oldfile	Delete one or more old files on the disk. DON'T DELETE ANY ".$$$" OR ".R FILES.
4.) EX	Continue finishing the edit session.

-- OR --

2.) EF	Save whatever was already written to disk. We will call this Part 1.
3.) EW a:part2	Setup to write the rest of the text to the file 'part2' on another disk drive, in this case drive "A:".
4.) EX	Write the rest of the text out to 'a:part2', and exit VEDIT.

You will now have to concatenate the two parts on the two disks back into one file. See "Concatenating Two Files" discussed previously.

Note: See the section "Disk Write Error Recovery" if you do not have enough space on any drive to save all of the file. The procedure becomes more complicated, but you can still save all of your changes and additions to the file.

Ending the Edit Session

To end the edit session and exit VEDIT, you must be in the command mode and issue one of the commands "EX" or "EQ". There is a world of difference between these two commands. "EX" is the normal command to end an edit session, and the text you were editing will be saved on disk. The "EQ" command, on the other hand, quits the edit session and DOES NOT save the text on disk.

The "EQ" command has several uses. One is to actually abort the edit session, because you didn't really want to modify the file at all or you made a big mistake in editing. Another use is for just viewing files - instead of using the CP/M (MSDOS) TYPE command, you can VEDIT the file. When you are finished examining the file, give the "EQ" command. Finally, if you save your file with the "EY" command or close your file with the "EF" command, you must use "EQ" to exit VEDIT.

Since this is just a practice session with VEDIT, the text you are currently editing is probably all butchered up and you don't want it saved on disk. Therefore the "EQ" command is the appropriate way to exit VEDIT now. Of course, if you would like to save your current text, you should exit with the normal "EX" command. If you give the "EQ" command, VEDIT will ask for verification before it actually aborts the edit session.

EXIT VEDIT TO CP/M:

1.) Exit Visual mode to Command mode.

VISUAL EXIT

2.) Exit Command mode to Operating System.

EX	Text is written out to disk and saved.
-- OR --	
EQ	Abort - This does not save the text on disk.
-- OR --	
EZ	Abort - like EQ, but stay in VEDIT.
-- OR --	
EY	Like EX, but stay in VEDIT.

This page reserved for your notes.

M E M O R Y A N D F I L E M A N A G E M E N T

Memory and File Management

This section covers the somewhat more technical topics of memory and file management by VEDIT. This includes a more detailed description of the text registers and an explanation of the automatic disk buffering used to handle large files. One additional file management feature that VEDIT provides is automatic disk searching for special files, the Auto-Startup file and the Help file. For most applications, it is not essential that you have a detailed knowledge of how VEDIT manages memory and large files. You may read this section simply to learn more about VEDIT, or in case you do require specific information.

The Text Registers

The ten text registers have two primary purposes. One is to hold sections of text which are to be moved or copied to other positions in the file being edited. The second is to hold command sequences which may be executed in Command Mode as macros. In either case, the registers are holding text; only the manner in which the text is used is different.

Generally, the text registers are all empty when VEDIT is first invoked. The registers are loaded by copying, moving or appending a portion of the main text to the register. This may be done in either Command or Visual Mode. Alternately, a register may be loaded directly from a disk file. The contents of a register may be viewed by typing it to the console; and it may be saved in a disk file. The text registers are not changed by any disk read/write operations. They can therefore be used to extract sections of text from one file and insert them anywhere in another file. Since inserting a text register does not destroy or change the register, it may be inserted repeatedly at different locations in the file.

When holding regular text, the registers act as scratch pad buffers in that they hold a temporary copy of text which is independent of the main text buffer. They serve the purpose of copying or moving sections or "blocks" of text from one area of the file to another, commonly referred to as "cut and paste" operations. Three operations are possible. One is to simply copy a section of the main text buffer to the register. The second is to move a section of text to the register, in which the section of text is also deleted from the main text buffer. For both the move and copy operations, the section of text can optionally be appended to any text which is already in the register. Third, the register contents can be inserted anywhere within the main text. The register may be inserted at the cursor position in Visual Mode or at the edit pointer in Command Mode.

Placing commands into the registers and executing these commands as "macros" is a very powerful facility. It is a useful method of saving long command sequences which must be executed repeatedly during

an edit session. If they are to be reused in the days ahead, they can even be saved on disk. Very sophisticated editing operations are also made possible. For example, say that you have a manuscript on disk as 20 different files and you find that you have consistently misspelled 40 words. This could be a very time consuming editing operation, but it can be greatly simplified with two command macros. One macro will contain the global search and replace for each of the 40 words. The second macro will contain the commands to edit each of the 20 files, and for each file execute the search/replace macro. Once the two macros are created, you execute the second macro and can take a coffee break while the 800 (20 times 40) operations are automatically performed. (ALWAYS make a backup copy of the files before performing complex macros. It is very easy for a small syntax error, or a power or hardware failure to destroy the files being automatically edited).

Since a text register can hold an entire file, it is possible to simultaneously edit the main file and up to three or four small files. Each small files is held in a text register, and a corresponding register is used to hold the command to "flip" between the main text and the small file. While the main file may be up to one disk in length, the small files must all fit into memory simultaneously.

Automatic Disk Buffering

Auto Disk Buffering refers to any disk file reading or writing which VEDIT performs automatically, without the user having given explicit read or write commands. (See also "Basic Editing Concepts".) The simplest auto disk buffering (called "Auto-Read") involves reading the input file into the text buffer when the editor is invoked in the normal way, and writing the output file when the editor is exited. VEDIT can also perform more sophisticated disk buffering when editing very large files. This can be done in either the forward direction, "Forward Disk Buffering", or in the backward direction, "Backward Disk Buffering". The following headings describe these two types of automatic disk buffering.

If the text buffer fills up in Visual Mode while the user is typing in more text, VEDIT will attempt to write out 1K byte sections from the beginning of the text buffer to the output file. This is referred to as "Auto-Write". If the 1K section of text cannot be written out, either because auto buffering is disabled, or because the cursor is positioned within it, VEDIT will display the message "FULL" on the status line. Text cannot be inserted until manual or automatic disk buffering is performed.

While it is most convenient to normally have auto-buffering enabled, there are times when an experienced user will want to disable it. This can be done from Command Mode with the "ES" command. If manual buffering is being done, automatic buffering could be an interference.

Forward Disk Buffering

When VEDIT edits a file it reads text from the Input file into the Text Buffer, where it is edited, and writes the edited text to the Output file. For a small text file, the operation is quite simple. The entire Input file is initially read into the text buffer for editing. When editing is complete, the text buffer is written to the Output file. In order to edit files which are too large to fit into memory all at one time (i.e. files which are larger than 40 Kbytes in a 64K system), the procedure becomes more complicated. Only a portion of the Input file is initially read into the text buffer for editing. In order to edit the rest of the file, some of the text buffer must be written to the Output file, and then more of the Input file read in for editing. This must be repeated until the entire file has been edited.

Conceptually, it might help to consider the displayed screen a "window" into the text buffer. This "window" may be readily moved anywhere within the text buffer with the [PAGE UP], [PAGE DOWN] and other cursor movement functions. Furthermore, the text buffer may be considered a "window" into the file. Moving this text buffer window toward the end of the file is referred to as "forward disk buffering", and moving it back toward the beginning of the file as "backward disk buffering".

Automatic forward disk buffering simplifies editing of large files. Forward disk buffering is performed in Visual Mode whenever the user reaches the end of the text buffer (by [PAGE DOWN], [ZEND], etc.), but the input file has not been completely read. It is almost invisible to the user, except for disk access time. VEDIT will then attempt to read more of the Input file and if necessary write text to the Output file. The minimum amount to be read from the Input file, "Minimum Transfer Kbytes", is determined during customization. If this much free memory is available, the Input file is read until the memory is "nearly" full. "Nearly" is defined as leaving the number of bytes free that you specified during customization. If this much free memory is not available, "Minimum Transfer Kbytes" will be written from the beginning of the text buffer to the Output file, before more of the Input file is read. See also Appendix A, "Memory Size Dependent Parameters".

Forward disk buffering is only done automatically in Visual Mode if it was enabled during customization or with the "ES" command (the command is "ES 2 1"). It should normally be enabled. The disk buffering may also be controlled manually in the Command Mode with the "A" and "W" commands. Knowledge of these commands is not necessary for most applications, since the automatic disk buffering accommodates most needs.

Auto-buffering is only performed in Command Mode for the global operations: _S, _F, _B, _Z, _L, and N, since it might otherwise interfere with special editing applications.

Backward Disk Buffering

VEDIT's backward disk buffering augments the forward disk buffering to further simplify the editing of large files. It can also be performed automatically in Visual Mode in such a way as to be almost invisible to the user (except for disk access time). However, for best results, it must be used with some care, because you are more likely to run out of disk space . Although VEDIT always lets you recover from running out of disk space, it is more complicated if you are using backward disk buffering.

Occasionally, you may want to edit some text which has already passed through the text buffer and has been written to the Output file. Without backward disk buffering, you would have to restart the edit session from the beginning (with the "EA" command). The backward disk buffering, however, lets VEDIT read text from the Output file back into the beginning of the text buffer for further editing. However, before reading text back from the Output file, VEDIT needs to make space free in the text buffer. VEDIT does this by writing text from the end of the text buffer out to a temporary disk file. (The file has a name extension of ".R".)

Backward disk buffering uses additional disk space to hold the temporary file. As the Output file is written, disk space is also used up. Reading from the Input file does not make free up any disk space, nor does reading back from the Output file. Without backward disk buffering, the maximum file size which may be edited is therefore 1/2 a disk, unless the Input and Output files are on different drives, in which case the maximum file size is a full disk. With backward disk buffering, the maximum file size is reduced to 1/3 a disk if everything is on the same drive, or else 1/2 a disk if the Output file is on another drive. The temporary file is always on the default drive. (With a 3 drive system you could safely edit a file one disk in length, by making the default, the Input and Output drives all different.) These file size limitations arise because in the worst case, VEDIT will need to create a temporary file which is nearly as large as the Output file, which is generally as large as the Input file.

If you use backward disk buffering and run out of disk space, you can still recover without loosing any edited text. The procedure is described in the Command Mode section under "Disk Write Error Recovery". To be on the safe side, unless you have a hard disk, we recommend that you customize VEDIT with backward disk buffering turned OFF and forward disk buffering turned ON. If while editing you decide you would like backward disk buffering, and are confident you have the disk space, you can turn it on the command:

```
ES 2 2
```

To calculate if you have enough disk space for backward disk buffering, use the MSDOS "DIR" command or the CP/M "STAT" command on the disk. If the amount of free space is twice the size of the file

you wish to edit, you are usually safe (unless the Output file will be significantly larger than the Input file). You can include any ".BAK" version of the file to be edited in the amount of free space available. If the amount of free space is not at least equal to the size of the file being edited, you will encounter a disk full error even without backward disk buffering. It is always best be to sure that there is enough free disk space before editing a file.

If you are at least two-thirds through a large file and wish to begin editing from the beginning again, it will generally be faster to restart the edit session (with the "EY" and "EB"commands), rather than using backward disk buffering.

Before you decide that backward disk buffering is never worthwhile, let us say that it is very useful with large capacity disk systems such as 8" double density or Hard disks where there is usually plenty of free disk space. If you are using a hard disk (such as on an IBM XT) it is suggested that you customize VEDIT with Backward disk buffering turned ON.

Automatic Startup

VEDIT will automatically execute a startup file on disk as a command macro. This can be used to setup various VEDIT parameters and to program the function keys on a CRT terminal. When invoked, VEDIT will attempt to read the file "VEDIT.INI" into text register 0, and then execute this register.

The file VEDIT.INI may contain EP, ES and ET commands to setup the various parameters, switches and tab positions. It may also contain an EB command which allows a particular file to be edited without it being specified when VEDIT is invoked. This may be handy if the same file is edited many times. The startup file may also contain the commands to load other text registers with text or commands from other disk files.

Some CRT terminals have programmable function keys which are initialized by sending (usually obscure) character strings to the terminal. The VEDIT startup file can perform this automatically. It is best done by loading the character strings into a second text register, typing out the register, and finally emptying the register. The CRT version of VEDIT comes with several example files for doing this.

To accommodate personal preferences and hardware configurations, it is possible to select on which drives VEDIT searches for the VEDIT.INI and the VHELP.TXT help files. The selections are made during customization. VEDIT can search on the default drive and on any specified drive for these files. On CP/M systems, VEDIT will search the default drive on the current user number and will search any specified drive on user 0. The auto-startup feature can also be turned off. See Appendix A for details.

Global File Operations

When editing very large files, the entire file will not all fit into the text buffer at one time. Since most commands and edit functions only operate on the text which is actually in the text buffer, editing a very large file becomes a little different from editing a smaller file. If VEDIT automatically allowed all commands to operate on the entire file, the entire editing process would slow down objectionably. Therefore, VEDIT allows you to specify an option on those commands which you may want to apply to the entire file. This option is referred to as the "global" option. When you specify the global option, VEDIT will automatically perform the disk buffering to treat a large file just in the same way as a small file (except for a little extra time). This option is available in Visual Mode for the [FIND] and [REPLACE] edit functions. The global option is selected in Command Mode by preceding the command with an "_" (underscore). The "_F" and "_S" allow Command Mode search and replace operation to be performed on a entire, arbitrarily large, file. The command "_B" allows direct access of the beginning of a file, and "_Z" direct access of the end of the file. The command "_L" allows you step through a file in Command Mode, with VEDIT performing the disk buffering when necessary.

Multi-Tasking Operating Systems

Some operation systems, such as Digital Research's MP/M and Concurrent, and Microsoft's XENIX, allow several programs to be run simultaneously on one computer system by one or more users. These operating systems must deal with the situation where one program attempts to access a file, which is already in use by another program. In effect, the second program is denied access to the file, or "locked out". This process is called "file locking". For example, two users cannot simultaneously run VEDIT on the same file.

VEDIT detects when it is running under these multi-tasking operating systems and then works in conjunction with their file locking. Typically if you try to access a file with VEDIT which is already in use by another program, the operating system will first issue you an error message. Then VEDIT will issue an additional error message (see Appendix C) and note that the file was not successfully accessed. You also cannot perform an "EC" (change disk) command on a disk which is in use by other programs. VEDIT will ensure that files which it is working on, or will soon need to access, are locked from use by other programs. VEDIT will also release files as soon as it is done with them so that they may then be used by other programs. (It closes all input files as soon as the end of the file is reached.)

This page reserved for your notes.

V I S U A L M O D E

Visual Mode

Screen Display

In Visual Mode the screen continuously displays a region of the file being edited and a cursor. The cursor indicates the exact location at which any edit changes, such as typing in new text, will be made. The bottom screen line, called the "Status line" displays information to help you in your editing. The rest of the screen displays text lines of the file being edited.

Text lines longer than the screen are handled in two ways. If horizontal scrolling is selected, long lines extend past the right edge of the screen and may be accessed by scrolling sideways. If horizontal scrolling is turned off, or if the line extends past the scrolling limit, it will be displayed on additional screen lines. These additional screen lines are called "continuation lines" and are indicated by placing a special "continuation character", in the leftmost column of the screen. The continuation character is selected during customization. A "-" is the default. The leftmost column of the screen is reserved for continuation characters. Due to technical reasons, the rightmost column of many screens is not used at all.

On CRT terminals the cursor is produced by the terminal and VEDIT can not change its appearance. However, on some systems (memory mapped), such as the IBM PC, VEDIT produces its own cursor and its appearance is user determined. You can choose an underline character, a solid block or a blinking block. Even the blink rate is determined. This is strictly a matter of personal preference. The options are more fully described under Customization in Appendix A.

VEDIT's interruptable screen updating allows the screen to be updated in the fastest way possible when you are performing rapid screen changes. You do not have to wait for the screen to finish updating before you continue editing. Operations such as [PAGE DOWN] require the entire screen to be updated. If you press another [PAGE DOWN] while the screen is being updated, VEDIT will interrupt the unwanted update and restart to display the most current screen. VEDIT, therefore, does not necessarily update the screen in the order in which you perform edit changes. It will skip the intermediate screen displays and go directly to the current screen display.

Status Line

The Visual Mode status line indicates what line number in the file the cursor is on, the cursor's column position, and what file is currently being edited. It also can contain words indicating conditions you should be aware of. The status line is also used to prompt you for a parameter, such as a search string or a text register number. (Some CRT displays allow the words and prompts to appear in reverse video.)

LINE AND COLUMN NUMBERS: The cursor's column position is the horizontal position on the current text line. The line number in the file is a count of the current number of preceding lines in the file, including any which have already been written out to disk. The line number for a particular line will therefore decrease if some of the preceding lines are deleted, and will increase if lines are inserted into the preceding text. These numbers are not updated immediately following every cursor movement, but only after you pause typing for about 1/2 of a second.

If desired, the display of either or both of these numbers may be turned off. This may be specified during customization or from Command Mode with the "EP" command.

FILE NAME : The name of the output file you are currently editing will be displayed in the middle of the status line. This is the name that the file will have when you save it on disk.

INSERT MESSAGE : When you are in INSERT mode a message appears on the status line to remind you of this condition.

1 END MESSAGE : Certain VEDIT functions require that you mark off the start and finish of a text block. When the first end is marked the message "1 END" will appear. Actually, it does not matter which function is used to set the first end, the function used to mark the second end will determine what is done with the block.

TEXT MESSAGE : When any text register contains text the message "TEXT" will appear (it will be overwritten by the "1 END" message).

FULL MESSAGE : If VEDIT runs out of memory space in Visual Mode, the message "FULL" will appear on the status line.

DISK MESSAGE : When VEDIT performs any automatic disk buffering in Visual Mode, it put the message "PLEASE WAIT FOR DISK" on the status line. This is also a reminder that your computer may not accept keyboard input during disk operations.

HORIZONTAL SCROLL OFFSET : When horizontal scrolling is being used and the left side of the screen does not display column 1 of the lines, the column number corresponding to the left side of the screen is display on the status line.

REPEAT COUNT : The currently selected repeat count when using the [REPEAT] function is displayed on the status line.

FUNCTION PROMPTS: Some VEDIT edit functions prompt you for a parameter, such a register or text marker number, or the search or replace string to be used. When prompted for a text register number, you may simply type the last function key again or [RETURN] to specify register "0". This is convenient if you only need one register for your text movement operations - you won't have to remember which register you saved the text in.

Keyboard Characters

Characters typed while in Visual Mode take effect immediately when typed. You can perform two basic kinds of operations: entering new text, or performing edit functions by typing control sequences. Newly entered text simply appears on the screen at the cursor position and is either inserted before, or overwrites the existing text.

Control sequences consist of either ASCII control characters, characters with the high order bit (Bit 8) set, or escape sequences. The customization process determines which edit function the control sequences perform. Unused control sequences can either be ignored in Visual Mode or inserted into the text. It is also possible to insert any control character into the text. The edit functions either move the cursor or perform a visual function.

Entering New Text

When a normal text character is typed, it appears on the screen at the current cursor position and the cursor then moves right. VEDIT has two modes for inserting new text, NORMAL and INSERT mode. When a text character is typed in NORMAL mode it appears at the cursor position and any character which was there is simply overwritten. In INSERT mode, no character is ever overwritten, but rather is squeezed to the right when a new character is typed at its position. In either mode, a new screen line, called a continuation line, is begun if the the text line becomes longer than the screen line. Visual functions exist to enter Insert Mode, revert to Normal Mode, or to switch between the modes. The mode which the editor starts in is set during customization and is a matter of personal preference.

Two convenient exceptions to the operation of Normal and Insert Mode pertain to the ends of lines and the RETURN key. Text typed at the end of a line is always inserted before the (invisible) <CR> <LF> pair, which ends each text line. Also, typing the RETURN key does not overwrite any character, but rather moves the rest of the line beginning with the character at the cursor position to a new text line.

The keyboard characters RETURN and TAB are normal text characters, but have special properties. The RETURN key causes a carriage return <CR> and line feed <LF> pair to be inserted into the text and a new line to be displayed on the screen. If it is typed while the cursor is pointing within a text line, that line is effectively split into two lines.

The [TAB CHARACTER] key causes insertion of a tab character, or optionally, spaces to the next tab position. The tab character itself is displayed with spaces on the screen to the next tab position, even though the spaces do not exist in the text buffer.

Any control characters in the text, other than <CR>, <LF> and

<TAB> are displayed in the common CP/M and MSDOS format by preceding the letter with a "Caret" (^). The edit function "[NEXT CHAR LITERAL]" allows any control character except <CTRL-Z> (which is not allowed by CP/M or MSDOS) to be inserted into the text. Alternately, the Command Mode "EI" command can be used to insert control and special characters which cannot be produced by the keyboard.

Performing Edit Functions

The edit functions fall into two categories: Cursor Movements and Visual Functions. The cursor movement keys only move the cursor to some other position in the text and do not actually change the text. The cursor may be moved forward and backward by a character, a word, a line, a paragraph or a screen at a time. The screen may be scrolled up, down, right and left. Up to ten positions in the text may be "remembered" with invisible markers which allow the cursor to be directly moved back to these positions. These and other movements are individually described later in this section.

Some of the visual functions perform delete operations, while others change the Insert mode, change the left margin, manipulate the text registers, and print text. Provided that the cursor has not been moved from the current line, the line can usually be restored to its original form (before any deletions or insertions were made) by using the [UNDO] function.

The particular control or function keys which perform the edit operations are assigned during customization. All versions of VEDIT have a default keyboard layout as described in the appendices. You can also create your own keyboard layout. When you are instructed, for example, to "press the [PAGE UP] key", you will need to refer to the appropriate keyboard layout sheet to see which control or function key on your computer is assigned to [PAGE UP].

The Repeat Function

Often it is desirable to repeat a typed character such as "*" or "-" when preparing tables, or to repeat an edit function such as [PAGE DOWN] in order to move quickly through the file. These can be performed with the [REPEAT] function.

When the [REPEAT] key is pressed, a "4" will appear in the left side of the status line. This is the repeat value. Pressing [REPEAT] again will increase the value to 16, and pressing it again a value of 64, and pressing it once more will give the maximum value of 256. If you want any other repeat value, you can simply type it in, i.e. "70". Allowable values are between 00 and 256. Once the repeat value is correct, simply type the displayable character or the edit function which is to be repeated. For example, to create the top of a box consisting of 50 "*", type [REPEAT], "50" and "*". Or to delete 16 lines type [REPEAT], [REPEAT], [ERASE LINE].

With VEDIT's interruptable screen updating, only the final screen will be shown when using the [REPEAT] key. Since some operations such as deleting a line may take a second to perform on a very large file, you may notice some delay when using the "Repeat" function. (If you are deleting many lines, it may be quicker to use the "K" command in Command Mode.) While the repeated function is being executed, the cursor will remain on the status line. If for some reason you want to abort the repeated function, press [CANCEL].

Horizontal Scrolling

VEDIT has the ability to scroll the screen sideways for editing documents with long lines, such as spreadsheets. Two edit functions are provided for scrolling the screen right and left: [SCROLL RIGHT] and [SCROLL LEFT]. The screen may be scrolled right up to a right margin called the "horizontal scroll margin" (which is independent of the word wrap margin). This scroll margin is user determined with a maximum value of 255. Lines longer than the scroll margin are continued on the following screen line.

The screen will also scroll automatically as you enter text or move the cursor. To reduce the amount of unwanted side to side scrolling, several functions such as [ZIP] and do not force the screen to scroll.

By setting a scroll margin of 78 when the screen width is 80 (the right and left most columns are reserved) the horizontal scrolling is effectively turned off and you can view an entire long line, since it will be displayed on multiple screen lines.

The default scroll margin is set during customization and may be changed in command mode with the command "EP 10 nnn".

Cursor Movement

A wide range of cursor movement functions is provided, including the four basic ones of [CURSOR UP], [CURSOR DOWN], [CURSOR LEFT] and [CURSOR RIGHT]. In general the cursor will only move to where there is text. That is, if the cursor is moved right past the end of a line it will move to the first position of the next line. This is always the case with left/right cursor movement, however, for convenience, VEDIT offers three different modes of up and down cursor movement.

MODE 0: the cursor can never be moved to a position that is past the end of a line. If you move the cursor down from the end of a long line to a shorter line, the cursor will also move left to the end of the shorter line.

MODE 1: the cursor can be positioned past the end of a line as the cursor is moved up and down. This mode enables you to move the cursor from a long line past short lines to another long line, staying in the

same column. If you attempt any edit change, i.e. typing in new text or deleting text, with the cursor past the end of a line, the cursor is first moved left to its "correct" position at the end of the line. Since this mode is especially important for horizontal scrolling, to prevent the screen from scrolling back as the cursor is moved past short lines, VEDIT automatically goes into MODE 1 when the screen is scrolled to the right.

MODE 2: the cursor moves identically to Mode 1. However, if the cursor is past the end of a line and you type text, spaces are automatically inserted from the end of the line up to the next text. This mode is handy for filling out tables and other formatted text. Note, however, that the many spaces will consume additional memory and disk space.

A little experimentation is best for understanding these modes and deciding which you like best. Mode 1 is the recommended setting. Other word processors generally operate in one of these three modes and you may want to pick that you are already familiar with.

The default mode is set during customization and may be changed with the EP command (example: "EP 9 0" sets cursor mode to MODE 0).

Setting and Jumping to Text Markers

You can place invisible markers in the text and later jump back to them. The positions are marked by typing the [SET TEXT MARKER] key. The status line will then prompt for a digit "0 - 9". Type a digit, or type RETURN or [SET TEXT MARKER] again to use marker "0". To move the cursor to a marked position type the [GOTO TEXT MARKER] key and the appropriate digit following the status line prompt. The cursor will then move to the marked position. You can abort either function by typing the DEL key or [CANCEL] function in response to the prompt for a digit.

The marked positions are relative to the text. This means that the markers will adjust themselves as text is inserted and deleted. All markers are initially set to the Home position. If text containing a marker is written to disk, that marker will be reset to the Home position. However, the RESTART function or an "EA" command will maintain the markers.

The Tab Character

One displayable character which acts a little different is the [TAB CHARACTER], which is normally assigned to the Tab Key or <CTRL-I>. When the Tab key is typed, it inserts the tab character into the text, which is displayed with spaces to the next tab position. The tab positions are variable, but are normally set to every 8 positions. You can tell the difference between the tab character and spaces by the way the cursor moves over them. The

cursor moves over each space individually, but moves over the Tab as a unit, i.e. a single [CURSOR RIGHT] might move you from column 1 to column 9. This reflects the fact that the Tab is a single character and should be treated as such. When the cursor is at the Tab character, it is displayed at the left side of the displayed spaces. If you wish to insert other characters before the Tab and leave the Tab in the file, you must be in INSERT mode. Otherwise the first character you type will overwrite the Tab and shift the rest of line left. The Tab character is commonly used when writing programs and aligning tabular data. Text paragraphs are best indented by not using a Tab, but rather by typing four or five spaces.

The [TAB CHARACTER] and the [TAB CURSOR] functions must not be confused. The latter is strictly a cursor movement function and has nothing to do with Tab characters. It only moves the cursor right to the character at the next tab position. It is very similar to typing [CURSOR RIGHT] repeatedly, except that it does not move the cursor past the end of the line. If you notice that you have customized the Tab key to be [TAB CURSOR] you are advised to change the Tab key to be [TAB CHARACTER] as it should be.

Optionally, the [TAB CHARACTER] function can insert spaces to the next tab position. This is equivalent to you typing in the spaces. While this uses up more disk space and is not normally recommended, it is useful in some applications. This option may be changed with the "ES" command (the command is "ES 1 1"). Although the screen display for these two options is identical, they are actually very different, especially to programs other than VEDIT.

If you set the tab positions in VEDIT to anything other than the default, you may find that other programs will not display your text as you wanted. This is due to your VEDIT tab positions being incompatible with the tab positions of the other programs, which usually have fixed tabs at every 8 positions. If you send text files with tab characters to a large mainframe computer, you may find that the tabs are lost in the transfer. (Many mainframes do not have tab characters internally.) These two cases are good candidates for allowing the Tab key to insert spaces to the next tab position.

Search and Replace

It is very easy to search for character strings and/or replace them with new strings in Visual Mode. A selective replace with prompting is also provided. Several search options may be selected.

FIND: To search for a string press the [FIND] function. The status line will prompt for the string. Enter the string you wish to find and type [RETURN]. VEDIT will move the cursor just past the first occurrence of that string. Each time you press [FIND] afterwards, VEDIT will move to the next occurrence of the same string. To search for a different string first press [CANCEL], which cancels the current search string. Pressing [FIND] after all the occurrences have been

found will result in a "CANNOT FIND string" message - type any character to continue. This also cancels the string, i.e. you don't have to press [CANCEL] to search for another string. If you need to search for a [RETURN] type <CTRL-N> in its place. The maximum string length is 31 characters. The [FIND] function is similar to the "F" command in Command Mode.

REPLACE: To replace one string with another press [REPLACE]. The prompt "FIND?" will appear in place of the status line. Enter the string to be replaced and press [RETURN]. The prompt then changes to "REPLACE WITH?". Enter the new string in the same fashion as the first. VEDIT will locate the first occurrence of the string to be replaced and prompt with:

REPLACE (Y / N / REST / CANCEL)?

Type "Y" to replace the string with the new string, or "N" or <SPACE BAR> to leave it unchanged. Type "R" to make the replacement and replace all subsequent occurrences without prompting for each one. As with [FIND], each time you press [REPLACE] VEDIT will locate the next occurrence of the string and prompt you with the replace options. To abort this process choose "C" from the options. Alternately, to search for and replace a different string, you can press [CANCEL] at any time.

Often you will want to search for all occurrences of a string and selectively replace some of them. This is done in conjunction with the [REPEAT] function. Press [REPEAT] four times for a maximum count of 256. Then press [REPLACE] and answer all prompts as above. Now the "REPLACE (Y / N / Rest / Cancel) ?" prompt will repeat up to 256 times allowing you to make the selective replace. As before, "R" will replace the remaining occurrences without prompting. Typing "C" will cancel the replace operation. To replace a new string with another, press [CANCEL] to cancel the previous replace operation.

To select any of the search options respond to the [FIND] or [REPLACE] "FIND?" prompt with an immediate RETURN. The status line prompt will change to:

OPTIONS (Begin / Global / Reuse)?

Select one or more options by typing the corresponding letters "B", "G" and "R", followed by a RETURN. For example, entering "BR <RETURN>" will select the "Begin" and "Reuse" options.

Begin This option will start the search or replace from the beginning of the text buffer. This is similar to first performing a [HOME]. If the "Begin" is preceded with the "Global" option, the search or replace will start at the very beginning of the file.

Global This option will cause the search or replace to operate to the

end of the file, if necessary, not just the end of the text buffer. If the entire file is in the text buffer, this option has no effect.

Reuse This option causes the previous search string or replace strings to be reused. This is convenient after you receive a "CANNOT FIND" message and want to reuse the strings on a new file or from the the beginning of the file (with the "Begin" option).

Following the options and the RETURN, you will get the "FIND?" prompt again, unless you selected the "Reuse" option. With the "Reuse" option the [FIND] or [REPLACE] will immediately be performed with the previously used strings.

Cancel Function

The [CANCEL] function cancels the current [FIND] and [REPLACE] settings and any function which is being performed because of a [REPEAT]. It also cancels functions which prompt you on the status line: [REPEAT], [COPY TO TEXT REGISTER], [MOVE TO TEXT REGISTER], [INSERT TEXT REGISTER], [SET TEXT MARKER] and [GOTO TEXT MARKER]).

The Text Registers

The most straight forward use of the text text registers is for cut and paste operations. They can hold up to ten sections, which have been "cut" and need to be "pasted" elsewhere.

The visual functions [COPY TO TEXT REGISTER] and [MOVE TO TEXT REGISTER] are used to copy or move text from the main text buffer to one of the text registers. The function [INSERT TEXT REGISTER] is then used to insert the contents of a text register at the cursor position. These functions are usually used to move or copy text from one area in the file to another. Text can also be moved to a text register, whose contents are then written to a disk file in Command Mode. The text registers used are the same as used in Command Mode, thus the text registers may be set in Command Mode and inserted in Visual Mode or vice versa.

To specify which text to move or copy, first position the cursor to the beginning of the text and mark it by pressing the appropriate function key. The message "1 END" will appear on the status line. Then move the cursor just past the last character of the text and press the function key again. The status line will then prompt for a digit "0 - 9" to specify which register to use. The digit may optionally be preceded with a "+" to indicate that the text is to be appended to any text which may already be in the register. You may type [RETURN] or the function key again in response to the prompt to use register "0". After typing the digit, the status line message will change to "TEXT". In the case of [COPY TO TEXT REGISTER], the

main text buffer will be unchanged, while in the case of [MOVE TO TEXT REGISTER], the text will be deleted from the main text buffer. Press [INSERT TEXT REGISTER] and a digit "0 - 9" to insert the specified register at the cursor position. Depending upon the "Point Past Register Insert" switch (see ES command), the cursor will be positioned either at the beginning or the end of the inserted text.

Whether the beginning or the end of the text is first marked is actually unimportant. It is also immaterial whether you press [COPY TO TEXT REGISTER], [MOVE TO TEXT REGISTER] or even [PRINT TEXT] when you mark the first end. Only when you mark the second end must the correct function be pressed. You can abort the operation by pressing the DEL key or [CANCEL] function in response to the status line prompt for a digit. If there is insufficient memory space for the text register, or to insert the register, the message "FULL" will appear on the status line and the no text will have been moved or copied.

The first marked end is invisible and you may forget where it is. To see where it is press [GOTO TEXT MARKER] followed by a special "+" and any digit. This operation will swap the positions of the marker and the cursor, placing the cursor where the marker was. Repeat the operation to restore the marker and the cursor to their original positions.

Indent and Undent Functions

As an aid in word processing and writing programs in structured languages such as Pascal, PL/I and C, the Visual Mode has the [INDENT] and [UNDENT] functions. These functions allow the editor to automatically pad up to the "Indent position" with tabs and spaces, when a new line is started with the RETURN key. The [INDENT] key moves the Indent position to the right by the "Indent increment", and the [UNDENT] key moves the Indent position back to the left. If the cursor is on a new line, or before any text on the line, when the [INDENT] or [UNDENT] is pressed, the cursor and any following text will also move to the new Indent position.

Normally the "Indent position" is zero and when a RETURN is typed, a <CR> <LF> pair is inserted into the text, and the cursor moves to column 1 of the next line. After the [INDENT] key is pressed once and a RETURN typed, the cursor will be positioned not in column 1, but rather at the first indent position, i.e., column 5 if the "Indent increment" is set to four. Pressing the [INDENT] key again will position the cursor still farther to the right after each RETURN, i.e., to column 9. Each time the the [UNDENT] key is pressed, the indent position moves back toward the left until it is back at zero.

The exact number of tabs and spaces inserted into the text buffer, to pad up to the "Indent position", is related to the currently set tab positions and the "Indent Increment". The padding will consist of the most tabs and fewest spaces in order to save memory and disk space. For example, assume that the "Indent

increment" is set to the common value of four (4) and the tab positions at every eight (8). When the "Indent position" is eight, the padding will consist of one tab; when the "Indent position" is twenty, the padding will consist of two tabs and four spaces. On the other hand, if the tab positions were set to every four, only tabs would be used in the padding. Note that if the "Expand Tab with spaces" switch is set, only spaces will be used for padding. This will use up lots memory and disk space.

Printing Text

VEDIT can print out any portion of the text which is currently in the text buffer. This can be done from both the Visual and the Command Modes of the editor. It is easiest to do in Visual Mode and is similar to the method of moving text to a text register. First position the cursor at one end of the text to be printed and press the [PRINT TEXT] key. (This is ESC - P in the example keyboard layout). Next, position the cursor at the other end of the text to be printed and press [PRINT TEXT] again, which causes the text to be printed.

To print the entire text move the cursor to the beginning and end of the text with the [HOME] and [ZEND] functions, and press [PRINT TEXT] at each end. The printing can be aborted by typing <CTRL-C>. Many printers use control characters or escape sequence to control such things as character size, font style and overstrike. These special control sequences can be imbedded directly into the text using the [NEXT CHAR LITERAL] function or the "EI" command. The Command Mode can also print out the contents of a text register.

Word Processing Functions

VEDIT has functions for moving the cursor to the beginning of the next word or the preceding word, and functions to delete the next or the previous word. The [NEXT WORD] function moves the cursor to the first letter of the next word. The [PREVIOUS WORD] function moves the cursor to the first letter of the current word, or if already there, to the beginning of the previous word. The [DEL NEXT WORD] function deletes the word, or portion of the word to the right of the cursor. [DEL PREVIOUS WORD] deletes the previous word and any following spaces if the cursor is at the beginning of a word. If the cursor is in the middle of a word, it deletes only that portion of the word to the left of the cursor. The delete-word functions never delete carriage returns, but rather just move over them when they are encountered.

Words are allowed to have imbedded periods in them, such as in "i.e.". A comma "," always ends a word, even if the comma is not followed by a space. As a special case, numbers with imbedded commas, such as "10,000" are treated as one word. The special characters ")", "]" and "}" also separate words from each other, as do spaces, tabs and carriage returns. All other characters are allowed in words.

The cursor can also be moved to the beginning of the previous or the next paragraph with the [PREVIOUS PARAGRAPH] and [NEXT PARAGRAPH] functions. VEDIT considers a paragraph to end when an empty line, a blank line or a print formatter command line is encountered. Print formatter command lines are considered to be any lines which begin with a ".", "!" or a "@" in the first column.

Word Wrap and Margins

To simplify word processing, the WORD WRAP facility may be used. This facility allows the user to specify a right margin beyond which no text should appear. If you attempt to enter new text beyond this margin, VEDIT will move the word which didn't fit within the margin to the next line, leave the cursor in the same position in the word, and add a carriage return to end the previous line. Word wrap will only occur when the cursor is past the right margin.

If you do not wish the text to begin in the left most column, you may set a left margin with the [INDENT] and [UNDENT] functions. A left hand margin may be set independent of whether word wrap is enabled. The right margin can be greater than the screen line length, in which case VEDIT will either scroll the screen horizontally or display a continuation line before the word wrap takes place. The word wrap facility is enabled by setting the right margin parameter. A value of zero turns word wrap off. This parameter is initially set during customization and can be changed with the "EP 7 nn" command. For example, to set the right margin at column 70, or to turn word wrap off, the following commands are given in Command Mode:

EP 7 70	Set word wrap at column 70
EP 7 0	Turn word wrap off

Formatting Paragraphs

In word processing it is frequently desirable to format a paragraph so that all of the text appears between selected left and right margins. The [FORMAT PARAGRAPH] function performs this. The left margin is set by the [INDENT] and [UNDENT] functions, while the right margin is the same as used for word wrap.

To format a paragraph, the cursor may be placed anywhere in the paragraph. After formatting, the cursor will be positioned at the beginning of the next paragraph. A series of paragraphs may therefore be formatted by just repeatedly pressing the [FORMAT PARAGRAPH] key. A paragraph will only be formatted if the right margin is greater than the left indent margin. Setting word wrap off also disables the formatting function.

When a paragraph is formatted, the additional indentation which the first line typically has will be preserved. If the second line of

the original paragraph is indented with respect to the first line, VEDIT will consider it to be an Offset paragraph and will also preserve this offset. Numbered paragraphs are often offset paragraphs as in the following example:

1.) This is an example of an offset paragraph. Notice how the first line begins in column 1, while all following lines begin in column 5. This paragraph can be entered by typing [INDENT] anywhere along the first line. Word wrap must of course be on.

However, any indentation which all lines of a paragraph have is ignored during by [FORMAT PARAGRAPH]. The left most line(s) will be positioned at the current left margin.

If you are using print formatter command lines, you should note that these lines separate paragraphs from each other and will not be formatted or changed in any way by the [FORMAT PARAGRAPH] function.

Lower and Upper Case Conversion

This is a relatively technical topic and is mostly applicable to users with non-standard keyboards and assembly language programmers.

Several modes are available for converting between lower and upper case letters as they are being typed on the keyboard. There are four options for converting from lower to upper case:

0.) No conversion is made.
1.) All lower case letters are converted to upper case.
2.) Conditional conversion of lower case to upper case for assembly language programming and other special applications.
3.) Similar to 2.) - upper and lower case letters are reversed

Mode "1" is similar to the "Caps Lock" on a keyboard, the 26 lower case letters are converted to upper case. Modes "2" and "3" are specifically designed for assembly language programming. In mode "2" lower case letters are converted to upper case if they occur to the left of a special character, called the "Conditional Convert Character", typically ";". To the right of the ";" they are not converted. In this manner an assembly language program can be entered or edited with all lower case letters and VEDIT will automatically convert the labels, opcodes and operands to upper case, while leaving the comment fields alone. This can also be used for FORTRAN programs and other special applications. Mode "3" is almost identical to mode "2"; instead of converting lower case to upper case, it reverses the case of letters appearing before the ";". This mode makes it easier to enter lower case literals into a program. These options and the special character are set with the "EP" command.

Upper and lower case letters can also be unconditionally reversed, i.e., lower case converted to upper case and upper case converted to lower case. This is specifically designed for the Radio

Shack TRS-80 Model I, whose keyboard normally produces upper case letters and lower case with the Shift key. This reversal is done immediately when a keyboard character is received and before any resulting lower case letter is converted to upper case as described above. The letters are also reversed for the Command Mode. This mode may also be handy in the case where most text is to be entered in upper case, but where an occasional lower case character is also needed. This mode is selected with the "ES" command.

End of Lines

Each text line is assumed to end in a <CR> <LF> pair as is required for other CP/M and MSDOS programs, and the <LF> is the true terminator of text lines. Typing the RETURN (or <CR>) key inserts a <CR> <LF> pair at the cursor position. Deleting the end of a line, will delete both the <CR> and the <LF>. Although VEDIT, in Visual Mode, will never create a line ending in just a <CR> or <LF>, such lines are handled in Visual Mode, although displayed differently. (Such lines can be created in Command Mode). If a line ends in only a <LF>, the next line will be displayed with a starting position directly below the end of the previous line. If a line contains a <CR> not followed by a <LF>, the <CR> will be displayed in the normal control character convention as "^M". Such lines may be corrected by deleting the offending lone <CR> or <LF> with the <DEL> key and then inserting the <CR> <LF> pair with the RETURN key.

Inserting Control Characters

Virtually any control character and non-ASCII character can be inserted into the text. These may be special printer control characters, the <ESC> character, or control characters for other purposes. Only <CTRL-Z> cannot be inserted because it is used by CP/M and MSDOS to mark the end of a file. The [NEXT CHAR LITERAL] function places the next character typed on the keyboard into the text. Any control character which can be generated from the keyboard can thereby be placed into the text. In case a character must be inserted which cannot be generated from the keyboard, the Command Mode "EI" command can be used. This command can insert any character with a decimal value between 00 and 255 (except 26 which is a <CTRL-Z>) into the text.

Special Character (High Bit) Support

This is a very technical topic and can be ignored my most users with CRT terminals.

Some systems such as the IBM PC and NEC APC allow accessing of special character fonts by using text characters which have their "High" or 8th bit set. These characters have a numeric value between 128 and 255, and VEDIT can be configured to properly display them.

Since many machines, particularly CRT terminals, do not support special character sets, VEDIT can alternately be configured to display High bit characters by stripping their High bit and displaying the resulting character in reverse video.

Most keyboards send only ASCII characters without using the High bit. Others, such as the IBM PC and NEC APC have function keys which send special characters consisting of all 8 bits. VEDIT is normally configured to decode all 8 bits, allowing the use of function keys for its editing operations. However, with some CRT terminals, the High bit is used as a "parity" bit which should be ignored. In this case, VEDIT can alternately be configured to ignore the High bit on keyboard input.

VEDIT's edit functions are accessed by typing control characters or function keys. The later either send a special High bit character or an Escape sequence. Those control codes which are not assigned to an edit function are normally ignored by VEDIT. However, for applications where special characters are to be entered directly into the text, VEDIT can be configured to enter unused control sequences into the text. Escape sequences are inserted as the corresponding character with its High bit set. Only those control sequences which are not used in the keyboard layout can be entered. Other characters can be entered with the "EI" command,

All of these options pertaining to special characters are controlled by the "EP 8 n" command. The default value is set during customization. The "EP 8 n" parameter is actually three parameters. The first bit enables High bit keyboard characters, the second bit enables display of special characters, and the third bit enables insertion of unused control codes. There are eight possible values, 0 - 7, each representing a combination of three bit values. The eight values and their meaning are:

VALUE	ALLOW HIGH BIT ON KEYBOARD INPUT	SPECIAL CHARACTERS OR REVERSE VIDEO	INSERT UNUSED CONTROL SEQUENCES
0	NO	REVERSE	NO
1	YES	REVERSE	NO
2	NO	SPECIAL	NO
3	YES	SPECIAL	NO
4	NO	REVERSE	YES
5	YES	REVERSE	YES
6	NO	SPECIAL	YES
7	YES	SPECIAL	YES

The normal value for CRT terminals is "1" - allow 8 bits on keyboard input, display High bit characters in reverse video (if possible) and ignore unused control sequences. The normal value for the IBM PC, NEC APC and other systems with special character fonts is

a "3".

Whether High bit characters are allowed on keyboard input has NO BEARING on High bit characters already in the text file. Such characters are left unmodified when they are read from disk or written to disk. If you wish to strip High bit characters in the text file (for example WordStar files), you can do so with the "YS" command.

Disk Buffering in Visual Mode

In Visual Mode, the disk buffering can perform automatic Read and Write to handle files which are larger than the size of available main memory. Specifically, if the current screen display reaches the end of the text buffer, and the entire input file has not yet been read, Forward Disk Buffering is performed.

Auto-buffering in the backward direction is performed when the cursor is at the beginning of the text and you press [HOME]. Therefore, pressing [HOME] [HOME] from anywhere in the text will perform backward disk buffering, reading back text which has already been written to the Output file. Nothing will happen if there is no text to read back in, or if backward disk buffering was not enabled during customization or with the "ES" command. The amount of text that VEDIT will buffer each time is set by Task 6.2 during the customization. This is generally between 4 and 14 Kbytes. To move back further in the file, just repeat the [HOME] function.

VEDIT will also begin to write out the text buffer (auto-write) if the memory becomes full while you are typing in more text. At this point the first 1K text bytes will attempt to be written to the output file. If no output file is open, or if the cursor is within the first 1K of the text buffer no writing occurs and the "FULL" message appears instead on the status line.

Both the auto-buffering and the auto-write may be disabled by the "Auto Buffering in Visual Mode" switch. Switch (2) of the "ES" commands controls whether auto buffering is enabled in visual mode. A value of "0" turns all auto buffering off. A value of "1" enables only auto buffering in the forward direction, and a value of "2" enables auto buffering both forwards and backwards.

Visual Mode Edit Functions

[HOME] Move the cursor to the very first character in the text buffer.

[ZEND] Move the cursor to the very last character in the text buffer.

[CURSOR UP] Move the cursor up one line, to the same horizontal position if possible. If the position is beyond the end of the line, move to the end of the line, if the position is in the middle of a tab, move to the end of the tab. If there is no line, it won't move.

[CURSOR DOWN] Move the cursor down one line, to the same horizontal position if possible. The same rules as for [CURSOR UP] apply.

[CURSOR RIGHT] Move the cursor to the next character in the text. If currently at end of line, move to beginning of next line. If there is no line, don't move.

[CURSOR LEFT] Move the cursor to the previous character in the text. If currently at beginning of line, move to end of previous line. If there is no line, don't move.

[BACK TAB] Move the cursor to the first position in the current screen line. If cursor is already at the first position, move to beginning of previous screen line.

[TAB CURSOR] Move the cursor to the character at the next tab position. If cursor is at the end of a line, don't move. Note that this only moves the cursor, use the [TAB] key to insert a Tab character.

[ZIP] Move the cursor to the end of the text line the cursor is on. If it already is at the end of a line, it moves to the end of the next text line.

[LINE TOGGLE] Is a combination of [ZIP] and [BACK TAB]. First moves the cursor to the end of the text line. If it already is at the end of a line, it moves to the beginning of the screen line.

[NEXTLINE] Move the cursor to the beginning of next text line.

[SCROLL UP] Similar to [CURSOR UP], except that the cursor remains on the same screen line and the screen moves down instead.

[SCROLL DOWN] Similar to [CURSOR DOWN], except that the cursor remains on the same screen line and the screen moves up instead.

[SCROLL RIGHT] Move the screen window right in order to view long lines going off the right side of the screen.

[SCROLL LEFT] Move the screen window left in order to view the beginning part of long lines

[PREVIOUS WORD] Move the cursor to the first character of the current word, or if already there, to the beginning of the previous word.

[NEXT WORD] Move the cursor to the first character of next word.

[PREVIOUS PARA] Move the cursor to be beginning of the current paragraph, or if already there, to the beginning of the previous paragraph.

[NEXT PARA] Move the cursor to the beginning of next paragraph.

[PAGE UP] This scrolls the screen to give a similar effect to typing [CURSOR UP] for 3/4 screen lines.

[PAGE DOWN] This scrolls the screen to give a similar effect to typing [CURSOR DOWN] for 3/4 screen lines.

[SCREEN TOGGLE] Move the cursor first to the last allowed screen line, or if already there, to the first allowed screen line.

[SET TEXT MARKER] Followed by a digit "0 - 9". Sets an invisible text marker which will automatically adjust as text is inserted and deleted.

[GOTO TEXT MARKER] Followed by a digit "0 - 9". Moves the cursor to the beginning of the line containing the specified text marker. If the marker has not been set or has been reset, moves the cursor home.

[SET INSERT MODE] Change the mode to INSERT if not already there.

[RESET INS MODE] Change the mode to NORMAL if not already there.

[SWITCH INS MODE] Switch the mode to the opposite. Note that normally either [SET INS MODE] and [RESET INS MODE] or [SWITCH INS MODE] would be implemented during the VEDIT Customization process.

[DELETE] Delete the character at the cursor position. The cursor doesn't move. A lone <CR> or <LF> will also be deleted, but a <CR> <LF> pair will both be deleted as one.

[BACKSPACE] Move the cursor left and delete the character at that position.

[DEL PREVIOUS WORD] Delete the previous word and any following spaces if the cursor is at the beginning of a word. Otherwise delete only that portion to the left of the cursor.

[DEL NEXT WORD] Delete the entire word and any following spaces if the cursor is at the beginning of a word. Otherwise delete that portion of the word at and to the right of the cursor.

[EREOL] This deletes all characters from the cursor position to the end of the text line but not the final <CR><LF> pair unless the text line only consists of the <CR><LF>, in which case the <CR><LF> is deleted. For example, the following sequence will delete an entire line:

[BACK TAB] [EREOL] [EREOL].

[ERLINE] This deletes the entire text line. Use of [BACK TAB] [EREOL] is actually preferable, since the latter does not close up the screen line and frequently allows the [UNDO] to restore the original line.

[UNDO] This rewrites the screen and ignores the changes made to the text line the cursor is on.

[NEXT CHAR LITERAL] The next character, whether a displayable character, a control character, or a character with its high order bit set, will be placed into the text buffer.

[REPEAT] The next text character or edit function is repeated. This is either a multiple of 4 or a number typed in. Type "000" and any character to abort.

[INDENT] This increases the "Indent Position" by the amount of the "Indent Increment". The editor will then automatically pad with tabs and spaces to the Indent position following each RETURN. The padding will also take place on the current line if the cursor is before any text on the line.

[UNDENT] This decreases the "Indent Position" by the amount of the "Indent Increment", until it is zero. One [UNDENT] therefore effectively cancels one [INDENT].

[COPY TO TEXT REG] The first time this key is hit, the position of the cursor is (invisibly) marked, and the message "1 END" is displayed on the status line. When the key is hit while the "1 END" is set, the status line prompts for a digit "0 - 9" indicating the text register to be used. The text block between the marked position and the current cursor position is then copied to the text register. Optionally the digit may be preceded with a "+" to indicate that the text is to be appended to any text already in the register. Assuming there is enough memory space for this "copy", the message "TEXT" is then displayed on the status line in place of the "1 END". If insufficient memory space exists, no copy is made, the "1 END" is erased and the "FULL" message appears on the status line. Hitting this key twice at the same cursor position will empty the specified text register. Note that either the beginning or the end of the text block may be set first.

[MOVE TO TEXT REG] This is similar to [COPY TO TEXT REG], except that the text block is deleted from the text buffer after it is moved to the text register.

[INSERT TEXT REG] Followed by a digit "0 - 9" indicating which text register's contents are to be inserted at the current cursor position. The register itself is not changed. If there is insufficient memory space for the entire "copy", nothing is inserted and the "FULL" message will appear on the status line. Moving the cursor to another line will clear the "FULL" message.

[PRINT TEXT] This is activated similar to the [COPY TO TEXT REG], only no digit needs to be typed. The block of text is then printed on the CP/M listing device. A <CTRL-C> will abort the print out.

[FIND] Performs a search operation. Prompts for the search string on status line. String may be up to 30 characters long and ends in RETURN. Use <CTRL-N> to search for RETURN. Press [FIND] again to search for the next occurrence.

[REPLACE] Performs a selective replace. Prompts for string to be replaced, then prompts for replacement string. Next, each time [REPLACE] is used, the next occurrence of the string is found and the user has the option of replacing it or not, and of replacing all further occurrences of the string or canceling the assignment.

[CANCEL] [FIND] and [REPLACE] reuse the previous strings until they are canceled with [CANCEL] or the search string is not found. [CANCEL] will also abort any function taking place because of a [REPEAT] and will cancel any function prompting on the status line for a text register/marker number.

[FORMAT PARAGRAPH] This will format the paragraph that the cursor is in so that all of the text appears between a left and right margin. The left margin is the current Indent Position, and the right margin is the current Word Wrap column. Following the format, the cursor will be positioned at the beginning of the next paragraph. Text output processors commands will not be formatted.

[VISUAL EXIT] Visual Mode is exited to Command Mode. The current cursor position in the text buffer will become the Command Mode edit pointer position. Any text register is preserved. Depending upon the value of the "Clear screen on visual exit" switch, the command prompt will appear either on a clear screen or just below the status line.

[VISUAL ESCAPE] This is identical to the [VISUAL EXIT], except that any current iteration macro is aborted.

[RESTART] The text buffer and any unappended portion of the input file is written to the output file. The output file is closed and then reopened as the Input and Output file. The file is then read into the text buffer again.

C O M M A N D M O D E

Command Mode

Command Mode Notation

$	denotes the <ESC> control character. Wherever "$" appears in a command mode example, type the <ESC> key.
<TAB> <CR>	represents the TAB character while <CR> represents RETURN. Press the <TAB> key or the RETURN key.
<ESC> <CTRL- >	represents the ESC key or alternate command mode escape character selected during customization. Other control characters produced by holding the CTRL key and typing a letter are represented by "<CTRL-letter>".
[ ]	The bracket characters used for iteration macros are printed as "[" and "]" in this manual. Some users may be more familiar with the angle brackets "<" and ">". You can choose which characters to use during customization.

Command Lines

In command mode the user enters command lines, which consist of single commands, concatenated commands or iteration macros. Each command line is ended by typing RETURN or the ESC key twice, at which point the command line is executed. The <ESC> is also used to delimit search strings and file names. In the event that your keyboard does not have an ESC key, you may customize the command mode escape character to be any other control character.

The ESC key is echoed with a "$", which is also used in the examples in this manual to represent the ESC key. The RETURN or <CR> key is echoed with a <CR> <LF> pair, and the pair is also entered into the command line.

The user is prompted for a new command line by the "COMMAND:" prompt. Before the command line is ended, the line may be edited with most common line editing characters. They are described in detail below. Once execution begins, it may often be aborted by typing <CTRL-C>. This causes a *BREAK* and a new command prompt to be displayed. VEDIT checks for the <CTRL-C> before any new command is executed, during the execution of the "A", "F", "N" and "T" commands, and in a few other situations.

Commands such as "I", "F" and "S", which take "text string" arguments must end in the "string delimiter", typically <ESC>. If you type RETURN before the final delimiter, the command prompt changes to "-" as a reminder.

Prompt: "-" This means VEDIT is waiting for the string delimiter

If you have made a mistake, receive the "-" prompt and don't know what the delimiter is, type <CTRL-C> to abort the command.

If, while typing, the command line should exhaust the amount of memory space available to it, (the text buffer, text registers and command line all share the same memory space) VEDIT will send the "Bell" character to the console and neither accept nor echo any more characters. The user will have to edit the current command line in order to end it and should then rectify the full memory situation. Even when the memory is full, (see "U" command) up to ten characters may be typed on the command line.

Command Line Editing

Several common control characters are recognized in command mode as line editing characters. They are:

<CTRL-H> or <BACKSPACE>	Delete the last character typed and echo a <CTRL-H> to the console.
<RUBOUT> or <DELETE>	Delete the last character typed and echo the deleted character to the console.
<CTRL-R>	Doesn't change the command line, but echoes the entire command line back to the console.
<CTRL-U>	Delete the entire command line and send a "#" to the console.
<CTRL-X>	Identical to <CTRL-U>.

If you wish to search for one of these characters in the text, or use one within any other string, you must precede it with a <CTRL-Q>. <CTRL-Q> causes the following character to be taken literally, and not be interpreted as a line editing character, a RETURN or any other special character.

Help Command

VEDIT offers an on-line help facility. A command summary, a keyboard layout, and some additional information are stored in the disk file "VHELP.TXT". When the "H" command is given, VEDIT will display this file, one screen at a time. The "VHELP.TXT" file should not be erased or renamed. However, it may be edited for your own needs.

The first time the "H" command is given, there will be a slight delay as VEDIT searches for "VHELP.TXT" on disk, and moves it to memory. The next time you type the "H" command, the help text will

appear immediately, since it remains in memory. Sometimes this is bothersome, since it uses valuable memory space. It can be removed from memory with the "-H" command.

Command: H <RETURN> - Begins displaying "VHELP.TXT" on console.

Type: RETURN - To see the next screen

Type: <CTRL-S> - Stops displaying "VHELP.TXT"; press RETURN key to continue.

Type: <CTRL-C> - Return to command mode.

Command: -H <RETURN> - Remove "VHELP.TXT" from memory.

"VHELP.TXT" is designed for screens 24 lines deep by 80 characters wide. Each screen is separated by a <CTRL-S> "stop character" which will cause the "H" command to pause before going on to the next screen. If your screen is smaller, you may need to rearrange some of the <CTRL-S> characters in the text. You can edit "VHELP.TXT" as you would any other file. The <CTRL-S> will display in Visual Mode as "^S".

We recommend that you change VHELP.TXT as soon as you are comfortable with VEDIT. You should change the keyboard layout to reflect any changes you made and you may want to add the file KEYS.ART to the help text.

On the technical side, the help text is loaded into a special hidden text register. The help text is then displayed using the "RT" command which pauses when a <CTRL-S> is encountered. Be careful when printing files which contain <CTRL-S>, since it may have an unexpected effect on the printer.

Controlling Console Display

Any screen output in Command Mode from commands such as "T", "RT" and "ED" can be stopped by typing <CTRL-S>. Typing any other key, but typically another <CTRL-S>, will then resume the screen output.

The "RT" command will also respond to <CTRL-S> characters that are in the text. The help file, "VHELP.TXT", is structured by incorporating <CTRL-S> characters within it.

FILE OPERATIONS

Exiting With Saving

Usually when you are done editing a file you will want to save the new file on disk and leave VEDIT. This is done with the "EX" command from Command Mode. It does not matter where you are in the file, or how large the file is, "EX" will save the file on disk and leave VEDIT.

Command: EX Save file on disk and leave VEDIT

It is also possible to save the file on disk and remain in VEDIT with the "EY" command. The "EY" command is useful when you are finished editing one file and want to edit another file. An added benefit is that the "EY" command does not alter the text registers.

Command: EY Save file on disk and remain in VEDIT

Quitting Without Saving

VEDIT allows you to quit editing a file without saving the changes. The "EQ" command will quit and return you to the operating system, while the "EZ" command will leave you in VEDIT command mode with no open files. Both commands require confirmation of the decision to quit (this is to ensure that you do not forget to save important changes). You can skip the confirmation prompt by including a "Y" in the quit command: "EQY" or "EZY". The text registers will retain their contents when using the "EZ" or "EZY" command.

Save File and Continue Editing

If you are making a lot of time consuming changes to a file, it is a good idea to routinely save the file on disk and then continue editing it. Otherwise all of your edit changes could be lost should a power or hardware failure occur. This also protects you from your own mistakes. The command to save a file on disk and then continue editing it is:

Command: EA Save file on disk and continue editing it

Following the "EA" command the edit pointer, or cursor in visual mode, will be at their previous positions. The text marker positions will also be maintained. It is frequently easier to use the [RESTART] function in visual mode rather than the "EA" command.

Editing a Second File

When you are done editing one file and need to edit another, it is not necessary to exit VEDIT and then re-invoke VEDIT for the second file. The "EY" command makes it easy to save the file being edited on disk, in preparation for editing another file. It performs the equivalent of the "EX" command without leaving VEDIT. The command to finish editing a file and begin editing another file (NEWFILE.TXT) is:

```
EY EB newfile.txt
```

Notice that spaces may be added between commands and in front of a file name to improve readability.

Directory Display

The very convenient "ED" command gives a display of the disk directory. Drive specifiers and the wildcard characters "?" and "*" can also be used. Some examples are:

ED	gives the directory of the default drive.
ED A:	gives the directory of drive "A".
ED *.ASM	gives the directory of all ".ASM" files.

The "ED" command serves as a reminder of what files on a disk have been or still need to be edited, merged split, etc. In case the disk becomes full and VEDIT gives you a "NO DISK SPACE" or "NO DIR SPACE" error, the "ED" command helps you delete any unneeded files on the disk. Files can be deleted with the "EK" command.

Note that with CP/M, an optional user number may be specified following any file name specified in VEDIT. Therefore, the command to display the CP/M directory of drive "B", user number "4" is:

ED B:=4	gives the directory of drive "B", user number "4".

NOTE: WHEN DELETING FILES, DO NOT DELETE ANY ".$$$" or ".R" FILES FROM WITHIN VEDIT! THESE ARE THE TEMPORARY EDIT FILES VEDIT IS USING. DELETING THESE FILES WILL RESULT IN LOST TEXT. THE VEDIT.COM OR VEDIT.CMD FILE CAN BE DELETED IF NECESSARY.

Extracting Portions of Other Files

It is possible to append another file to the end of file being edited with the "ER" command, in effect merging the two files. In many instances, however, you will want to extract only a portion of another file and insert it into the middle of the file being edited. This portion might be a paragraph, just a single sentence or a programmer's subroutine. This is done with the "EG" command. The "EG" command takes as arguments the file name and the line range to be inserted. Therefore, you will need to know the line numbers of the text to be extracted. This can be determined with the "EL" command:

Example: EL MYFILE.TXT View MYFILE.TXT with line numbers

Simply note the beginning and ending line numbers of the text to extract. Lets say the line numbers are 235 and 272 respectively. Then the command to extract the text would be:

Example: EG MYFILE.TXT[235,272]

The text will be inserted at the edit position.

If you don't want to view all the lines of a file you can issue the "EL" command with a beginning and ending line number as in the "EG" command. For example: "EL MYFILE.TXT[200,400]".

The "EG" and "EL" commands allow a drive designator to be included. With CP/M systems, a User number may also be specified.

Disk Buffering in Command Mode

While the disk buffering can be fully automatic in Visual mode, it is not done automatically in command mode because it would interfere with the explicit file handling often done in command mode. VEDIT has a full set of commands for reading and writing files. Commands must be issued in order to read a file, write a file and perform forward disk buffering. In some cases it will be easier to switch into visual mode and allow it to perform the disk buffering automatically.

The "ER" command opens a file for reading, but does not actually read anything in. The file can be read with the "nA" command. Similarly, the "EW" command opens a file for writing, but does not write anything out. Text can be written out with the "nW" command. Forward disk buffering in command mode, therefore, requires successive "W" and "A" commands.

Some commands perform automatic reading/writing when invoked. The "EB" command performs an auto-read which reads in the entire file from disk or until the text buffer is nearly full. The "EY" command performs all the reading and writing to finish editing and saving a file without leaving the editor.

The commands which take the "global" modifier will also perform automatic disk buffering. The command "_B" moves back to the beginning of the file, performing backward disk buffering as needed. The command "_Z" moves the end of the file, and the command "_L" moves forward in the file, each performing forward disk buffering as needed. The "N" command (same as "_F") can perform forward disk buffering to find occurrences of a string anywhere in the file. These commands operate regardless of the setting of the "Auto Buffering in Visual Mode" switch.

As described earlier, backward disk buffering is accomplished by writing text from the end of the text buffer to the temporary ".R" file, and reading back text already written to the Output file. VEDIT can perform this disk buffering automatically in Visual Mode. The "-W" and "-A" commands allow you to do this manually in Command Mode. Because of the complexity of these commands, we suggest you not use them until you are thoroughly familiar with all other aspects of VEDIT's file handling. Generally, the easy to use "_B", "_Z", "_L"and "_F" commands can perform any additional disk buffering you will need in Command Mode.

The "-nA" and "-0A" commands allow text which has already been written to the Output file to be read back into the text buffer. "-nA" will read "n" lines back from the output file, or until the text buffer is full, or the output file is empty. "-0A" will read lines back until the text buffer is "nearly" full or the output file is empty.

The "-nW" command will write text from the end of the text buffer out to the temporary .R file. Its only purpose is to make more memory space available for performing the "-nA" command, or any other time you need more memory space.

Whenever an "A" command is issued or VEDIT performs auto disk buffering, VEDIT will always read the contents of the ".R" file back into the text buffer, before reading any more from the Input file. You, therefore, do not need to explicitly remember whether or not there is any text in the VEDIT.REV file.

Disk Write Error Recovery

Since many systems run with floppy disks which have limited storage capacity, the typical user will occasionally encounter a "Full Disk" error condition. This is caused by either running out of disk space, leading to the error message "NO DISK SPACE", or running out of directory space, leading to the error message "NO DIR SPACE". Fortunately, VEDIT allows you to recover from these errors using one of two recovery procedures. One is to delete files from the disk using the "EK" command until enough space exists to write the rest of file out. The second is to use the "EC" command to allow removing the full disk and inserting another disk on which to complete the

operation. The following paragraphs describe these procedures in some detail, and an example is given in the Tutorial.

The best policy is to avoid "Full Disk" errors by making sure that there is enough space before you begin editing. If you are editing files more than 1/3 disk in length, it is best to read the Input file from one drive and write the Output file on another drive. For example, if the Input file and VEDIT are on drive A and the disk in drive B is blank, give the command:

```
VEDIT infile.ext b:outfile.ext
```

The simplest and most common recovery is to delete files from the disk which is full. You can use the "ED" command to display a directory of the disk. If you find files which you can delete, you are all set. You can then re-issue the command which led to the full disk error. Any ".BAK" files can usually be deleted. You can also consider deleting any files which you know are backed-up on other disks. Never delete the ".$$$" and ".R" files from within VEDIT. (You can delete them from the operating system, in the unlikely event they appear on the directory there.)

If you are still reading this in order to learn more about VEDIT, STOP. You are very unlikely to ever require the following procedures. They are described here for completeness only.

There may be times when you cannot delete enough files to finish the edit session. You then have several alternatives. One is to close the current output file (with the "EF" command) and create a second output file on another drive. An example is:

```
EF              Close the current output file.
EWA:PART2       Create an output file on another drive.
EX              Exit the edit session.
```

You can then use VEDIT (or CP/M PIP or MSDOS COPY) to merge the two partial output files back into one file. (See the Tutorial for merging files.)

If all the disks in the drives are full, you will have to either change disks using the "EC" command or delete the Input file. In either case you want to read as much of the Input file and hope that there is enough room to read all of it. Begin by issuing the command:

```
#A
```

Then look at the end of the text buffer to see if all of the file was read in. If not, the recovery will be more complicated. If all of the Input file has been read, it is often simplest to delete this Input file from disk with the "EK" command. This will make enough space available for the rest of the Output file. If you delete the Input file, there will be no ".BAK" backup file when you exit VEDIT. For example, if the file you are editing is "LETTER.TXT", you could

give the following commands:

```
EKletter.txt      Delete the Input file.
EX                End the edit session.
```

Alternately, if you need to keep the original Input file, you can use the "EC" command to change disks and write the second part of the Output file to an empty disk. First issue the "EF" command to close the current Output file. The "EC" command will allow you to insert another (empty) disk into any drive. Example commands are:

```
EF                Close the current output file.
EC                and type RETURN
EWPART2           Create a second output file.
EX                End the edit session.
```

You will then have to use VEDIT (or CP/M PIP or MSDOS COPY) to merge your two Output files back into one file. This procedure has several potential shortcomings. If you were using backward disk buffering, you may get the error message "REV FILE OPEN", in which case you cannot change any disks. You will then have to make more space on the existing disks by deleting files, possibly the Input file.

If you were unable to read the entire Input file into the text buffer, the procedure becomes still more complicated. (Try again to make more space free on the existing disks!) If you have a copy of your Input file on a backup disk, delete the Input file, which should free enough disk space to end the edit session. All text which you just edited or entered will be in the Output file, but the Output file will be missing the last portion of the Input file which was never read in. You must examine the Output file to see how much is missing. Then copy your backup of the original Input file to a blank disk. Edit this file by deleting the entire front portion up to the text which is missing from the partial Output file. Exit VEDIT. Then use VEDIT or PIP to merge the Output file and the unread portion of the original Input file back together. This is a complicated procedure, but at least none of your edited text is lost.

If in the previous paragraph you know that you don't have a backup copy of the Input file, you will have to use the "EC" command procedure to write a second Output file to a blank diskette. However, by using the "EC" command, VEDIT will not be able to continue reading any unread portion of the Input file. You will therefore have to merge the Output file from the original disk with the second Output file, with the unread portion of the Input file.

If you cannot change disks because you were using backward disk buffering, you will have to make more space free on the existing disks by deleting files. If you want to avoid the complexities of deleting the Input file, you can delete any ".COM" or ".CMD" files, including VEDIT, which you can probably restore from a backup disk.

COMMAND MODE FEATURES

Search Options

There are three search options which can be selected by preceding the search command with a special character. The first specifies that the search or replace operation is to be performed in the rest of the file (global search or replace) instead of just to the end of the text buffer. The other two search options are useful for some applications, particularly when using command macros. One allows text strings to be delimited without using the <ESC> character. The second allows search error messages to be suppressed.

Search Option	Meaning	Affected Commands
_ (underscore)	Perform global search or replace	F, S
@	Use explicit text delimiter	F, N, S, I, RI, YT
:	Suppress search error messages	F, N, S

The "_" (underscore) which acts as the global search option, may also be specified with several other commands to make them "global" to the file, if necessary performing automatic disk buffering. You may want to think of the "_" character as a command option which makes the size of files less noticeable. A special character which may appear in some commands is <CTRL-Q>, which allows the following control character to be used literally. A second special character which may appear in a search commands is the pattern match code "¦". This is described fully in the next section.

Special Character	Meaning	Affected Commands
_ (underscrore)	Perform global operation	F, S, B, L, Z
<CRTL-Q>	Use next character literally	F, N, S, I, RI, YT
¦	Pattern match code	F, N, S

The commands "F", "N", "S", "I", "RI" and "YT" are followed by a string of characters called a "text string". Since the text string can be of any length and contain any character, including RETURN, there has to be some way of indicating the end of the text string. This is done with a special character called the "text delimiter", which is normally the <ESC> character. An option allows an "explicit delimiter" to begin and end the text string. With this option, the character immediately following the "F", "N", "S", "I", RI" or "YT" command is the delimiter. Any character can be the delimiter, but "/" is a good choice. Note that the text string itself cannot contain the explicit delimiter. This option can be invoked by preceding the command with a "@". In the following examples, the commands on the left side are equivalent to those on the right.

```
Fspeled$V                @F/speled/V

Sspeled$spelled$V        @S/speled/spelled/V
```

```
4Fpoint$V              4@F:point:

Ia new line$$          @I/a new line/
```

The explicit delimiter option can be made the default with the command "ES 9 1" or during customization. With the option ON, the "@" character is no longer needed. Although using this option requires more characters to be typed, many users find that it makes the commands more understandable. It eliminates the need for <ESC> to terminate any text strings and nearly eliminates the need for <ESC> entirely. It also allows the <ESC> character to be searched. For example, the following command searches for the string "h<ESC><ESC>":

```
@F/h<ESC><ESC>/V
```

Note that the <ESC> <ESC> therefore does not end a command if it appears before the explicit delimiter. If you type <ESC> <ESC> or RETURN before the explicit delimiter, the command prompt changes to "-" to remind you that VEDIT is still waiting for the delimiter. For example, to find "LABEL" at the beginning of a line, the search command would be:

```
F<RETURN>
LABEL<ESC><ESC>          Note: Prompt changes to "-" here
```

The "-" prompt is normal for strings which contain RETURN. However, if you get a "-" by mistake, type <CTRL-C> to abort the command.

The command "F$$" will always search for the last used string, even if the explicit delimiter was used for the original string or is currently in effect.

```
F$$      Search for last used string. (Must end in two <ESC>)
```

Search error messages can be suppressed by preceding the "F", "N" or "S" command with a ":". Alternately the suppression may be turned ON with the "ES" command or during customization. This is primarily used with command macros which contain many "S" commands, and where the macro should not terminate if some of the strings are not found.

The literal character <CTRL-Q> operates similar to the [NEXT CHAR LITERAL] in visual mode - the next character is treated literally and not interpreted. This is the only way to search for characters such as <CTRL-R>, <CTRL-U> and <CTRL-H> which are also used for line editing. It is also an alternate way to search for the <ESC> character. In the following examples, one command inserts a CTRL-H> into the text and the second command searches for a <CTRL-H> :

```
Iword<CTRL-Q><CTRL-H>$$        Insert a <CTRL-H>

Fword<CTRL-Q><CTRL-H>$$        Search for a <CTRL-H>
```

These two commands both search for the string "h<ESC>":

```
@F/h<ESC>/                    Fh<CTRL-Q><ESC>$$
```

CP/M, MSDOS and VEDIT all require that lines end in a <CR> <LF> pair. However, when files are transferred from mainframe computers, the lines often end in a <CR> without the <LF>. These lone <CR> must be changed to <CR> <LF> pairs. One cannot simply search for a <CR> by typing the RETURN key because it is expanded into <CR> <LF>, unless the RETURN is preceded with a "<CTRL-Q>". Therefore, the command to change all lone <CR> to <CR> <LF> pairs is:

```
b#S<CTRL-Q><CR>$<CR>$$        Change <CR> to <CR><LF>
```

Pattern Matching

This is a fairly advanced topic which you should be aware of, because of its potential usefulness and power. However, you should wait until you are familiar with VEDIT's basic search and replace operations before attempting to understand the details.

Often when performing a search operation, you want to search for a type of character, instead of a particular character. For example, you may want to search for the next number or for the next word beginning in a capital letter. VEDIT allows for a wide variety of such characters to be searched. Such searches may be made from the Command Mode or the Visual Mode. These categories of characters are called "patterns". Patterns can be searched with "Pattern Matching Codes". Each pattern matching code consists of a special "lead-in" character followed by a mnemonic letter. Normally the lead-in character is "¦". If your keyboard does not have this, it can be changed during customization. The mnemonic letter may be entered in upper or lower case. For purposes of clarity, all examples will show these letters in upper case.

The most commonly used "pattern" in a search is the "separator". A separator is any character which is not a letter or a numeric digit. For example, a simple search for the word "and" would result in matches by "sand", "Andres" and others. A search for " and " would be better, but would fail if the "and" appeared at the beginning or end of a line. The pattern match code for a separator is "¦S". The ¦S" will match a space, a tab, a RETURN or any other character which is not a letter or a digit. Therefore, the best search string for the word "and" is:

```
String:  ¦Sand¦S              Best search string for the word "and"
```

Another pattern is "¦X" which will match any character. There are many other patterns possible in VEDIT. For routine word processing, you will probably never need them. The pattern matching codes are:

|X Matches any character (equivalent to use of | alone in previous versions of VEDIT)
|A Matches any alphabetic letter, upper or lower case
|B Matches a blank - a space or a tab
|C Matches any control character - a character with a value of 0 - 31 (decimal)
|D Matches any numeric digit - "0" through "9"
|L Matches any line terminator - Line Feed, Form Feed or End Of File
|M Matches multiple characters - zero, one or more characters so that the string following the "|M" is satisfied
|N Matches any character which is NOT the following single character or pattern matching code
|R Matches any alphanumeric character - a letter or a digit
|S Matches any separator - a character which is not a letter or digit
|U Matches any upper case letter
|V Matches any lower case letter
|W Matches "white space" - one or multiple spaces and/or tabs
|| Matches a "|" - this is the literal "|". Actually, any undefined pattern matching code, will match a literal "|"

The |M and |N require a little explaining. The |N is a negation, similar to our usage of "Not". The string "|Na" (think of it as not "a") therefore matches any character except "a". The command:

```
Command: Fexam|Ns$$          ("$" designates the <ESC> key)
```

would find occurrences of "exam", "examiner", but not "exams".

The code "|M" is useful for finding patterns where the beginning and end are defined, but the middle doesn't matter. The string "|Sa|Mtion|S" matches words beginning in "a" and ending in "tion". Besides being useful in searches, the "|M" code can be used to delete large blocks of text. For example, the following command would delete this paragraph:

```
@S/ The code|Mparagraph://          End with a RETURN
```

This "S" command is using "explicit delimiters", i.e. the "/" characters. This allows the command to end in a RETURN instead of two <ESC>. The double "//" at the command end means that the text which is found will be replaced by nothing, i.e. deleted.

In assembly language programming, any text following a ";" character is considered a comment. Instructions are often followed by a few tabs (to align the comments), the ";" and the comment. The following command will delete the tabs (and/or spaces) and the comment which follows any instruction. Lines which are entirely comments are left unchanged.

```
#@S/|W;|M<RETURN>
/<RETURN>
/ This command strips comments
```

Iteration Macros

An iteration macro is a group of commands which will be repeated with or without user intervention as many times as desired. They are most useful in search and replace tasks (changing all instances of a misspelled word, for example).

An iteration macro's general construction is: a group of commands enclosed by brackets "[" and "]" and preceded by an iteration count which tells VEDIT how many times to repeat the commands between the brackets. The following example changes the first three occurrences (if found) of " teith" to " teeth".

Example: 3[S teith$ teeth$]

The iteration macro operates by executing the first command of the group through the last command, and then starting over again with the first command. The entire group will be executed the number of times specified by the iteration count.

It is very important to observe the placement of any necessary <ESC> to terminate strings and file names in iteration macros. File names must always be followed by an <ESC> and all text strings must be ended with an <ESC> unless you are using explicit delimiters. The following example shows a common error in which the string "enter me three times]" is entered into the text, which is not the intention.

Wrong: 3[I enter me three times]

Right: 3[I enter me three times$]

If desired, each command may be ended with one <ESC>, in which case you won't have to remember whether the command must be ended in an <ESC> or not.

The "[" and "]" may also occur within each other ("be nested") for more complicated commands. For example, the command "5[4T]" displays the same four lines five times for a total of 20 displayed lines. The command "3[5[4T] 4L]" will display the same four lines five times, then move to the next four lines and display them five times and last, move to the next four lines and display them five times.

Iteration Counts:

If no explicit iteration count is given, it defaults to "#" (32767) which signifies "forever" or "all". This is used when the iteration is to continue as long as possible. "#" represents the maximum positive number 32767. For clarity sake, the "#" may also be explicitly specified. The following example changes all occurrences of " teith" to " teeth".

Example: #[S teith$ teeth$]$$

It is normal to get the error message "CANNOT FIND ..." when performing a search or replace command for all occurrences of a string, because the command is literally searching for 32767 occurrences. However, the error will not occur for the "#S" command.

Using Visual Mode in Iteration Macros:

Search and replace operations are often used in conjunction with the visual mode in order to edit the region, or to confirm that the replacement was done correctly. For example, the following command will search for all occurrences of the word "temporary" and let those regions of the text be edited in visual mode.

```
[Ftemporary$V]
```

The following command could be used in a form letter to change the string "-name-" to the desired name, check that it was done correctly in visual mode, and if necessary make any edit changes.

```
[S-name-$Mr. Jones$V]
```

The Visual Mode has two ways of exiting back to Command Mode in order to help in using iteration macros. The [VISUAL EXIT] simply exits and lets any command iteration continue. The second, [VISUAL ESCAPE] exits to Command Mode, but also aborts any iteration macro. The latter is used when the user realizes that the iteration macro is not doing what was intended and does not want the macro to further foul things up. For example, in order to change all occurrences of the word "and" to "or", the following command may have been given:

Wrong: [Sandorv]

The user might then see in Visual Mode that the word "sand" was changed to "sor", which was not the intention. The [VISUAL ESCAPE] would stop the command and the following correct command could then be given:

Right: [S and$ or$v]

If it is unnecessary or undesirable to view each substitution in Visual Mode, the previous replace operation could take the simpler form:

```
#S and$ or$$
```

Note that this is not an iteration macro, but rather just a form of the "S" command. Because it executes much quicker, it is preferable to the equivalent command:

```
Slow:          [S and$ or$]
```

The commands "I" for Insert and "T" for Type are useful in iteration macros. The "T" can be used to type out the lines that are changed in an iteration macro without going into Visual Mode. The "I" command is useful when the same text is to be inserted into the text buffer many times. For example, to begin creating a table of 60 lines, where each line begins with a <TAB> and ".....", the following command can be used before the rest of the table is filled in Visual Mode:

```
60[I<TAB>.....<RETURN>$]
```

The <RETURN> will be expanded into a <CR> <LF> pair.

Iteration macros begin operation from the current edit pointer position. Therefore, be sure to place the edit pointer correctly before executing an iteration macro.

An iteration will continue until its iteration count is exhausted or until an error occurs. A common error is an unsuccessful search operation and many iterations will normally stop with an unsuccessful search error message. A special case is using search commands ("F" and "S") in iteration macros with search errors suppressed. In this case, when a search is unsuccessful, no error is given, but the iteration is stopped, and execution continues with the command following the iteration. This may be an outer level iteration. Recall that the commands "#S" and "#F" are only unsuccessful if no occurrences are found.

Another special case occurs when using the "L" command in iteration macros. If the "L" command encounters either end of the text buffer, it too will stop the iteration with execution continuing with the command following the iteration. This is convenient for any iteration which needs to stop when the end of the text is reached. The supplied print macro "PRINT.EXC" is such an example.

Printing Text

Text can be printed from command mode with the "E0" command. This command takes a numeric argument similar to the "T" command to specify how many lines before or after the edit pointer are to be printed. For example, "40E0" will print the following 40 lines, while "-5E0" will print the preceding 5 lines. Additionally, the command "0E0" will print all lines from the beginning of the text buffer to the current edit pointer. (The edit pointer is the same as the cursor position when you change from visual to command mode). Therefore, the command to print the entire text is:

B#E0 Print entire text on line printer.

WordStar (TM) Files

WordStar (tm) files and files from other word processors often contain characters which have their High or 8th bit set. These are often difficult to edit with VEDIT and will often be displayed with reverse video characters. Such files can be converted to normal text files by using the "YS" command which "strips" the 8th bit. Its syntax is "mYS", where 'm' specifies the line range, as in the "T", "K" and "L" commands. "10YS" strips the High bit in all characters in the next 10 lines.

Command: B#YS Strips 8th bit of all characters in the text buffer.

Text Registers

Nine commands are available for using the ten text registers in command mode. Lines of text may be copied to a register with the "P" command:

35P5	Copy the next 35 lines to register 5.
-6P+4	Append previous 6 lines to register 4.
0P2	Empty out register 2.

The "G" command inserts the contents of the specified register at the edit pointer:

G2	Insert register 2 at edit pointer.

The "RS" command will save the contents of the specified register in a disk file. Various portions of a file or files may therefore be appended to a text register, which is then saved as a new disk file.

RS4 b:filesave.reg	Save contents of register 4 in "filesave.reg" on drive "B".

Similarly the "RL" command will load a register from a disk file. This can be useful for merging several files together in complex ways.

RL4 b:filesave.reg	Load register 4 from "filesave.reg" on drive "B".

The contents of a text register can be display on the console with the "RT" command:

RT9	Type out contents of register 9.

The "RT" command expands control characters, displays <ESC> with a "$" and pauses when a <CTRL-C> is encountered. Since this is not suitable for initializing a terminal (programmable function keys, etc.), the "RD" command is provided, which does not expand control characters:

RD9	Dump out contents of register 9.

The "RP" command prints the contents of a text register. This is useful for examining the contents of a text register. It also allows a disk file to be printed after first loading it into a text register.

The "RU" command displays the number of characters contained in each of the text registers. The sum of these ten values plus the number of bytes in the special "help" text register (used for the H command) is the last number displayed by the "U" command. If this sum

is not zero, the status line message "TEXT" appears in Visual Mode.

The "M" command executes the contents of the specified register as a command macro, as described in the following section.

Command Macros

The following pages describe the process of grouping commands together into text oriented "programs" called "Command Macros". It assumes that the reader is familiar with the individual commands of VEDIT. You should not feel compelled to understand this section until you are familiar with all other aspects of VEDIT.

The following pages also describe several additional commands in VEDIT. While these commands could be used by themselves, they are primarily intended for use in command macros and are therefore described in this context.

Command Macros

The text registers may hold commands which can be executed just as if they had been typed in by hand. Frequently used commands, particularly long iteration macros, can be saved in the text registers. These commands are referred to as "command macros" or just "macros" for short. The macros are usually created and edited in visual mode and are then moved to the appropriate register. The macros can also be saved on disk and be retrieved from disk (see RL and RS commands). Macros offer so many capabilities that it is impossible to cover all of the possibilities.

A macro is invoked with the "M" command:

```
M6          Executes macro in register 6.
```

A macro may contain an "M" command to invoke a macro in another register. This can be done to a depth of five.

A common use of macros is for large search and replace operations. Consider the example of a long manuscript split into 20 files in which 40 words were consistently misspelled. The task of correcting the words in all 20 files can be done with 2 macros. One will contain the search and replace for the 40 words. The second will edit each file, and for each file execute the search/replace macro. The first macro would appear as:

```
ES 8 1
ES 9 1
b#s/word1/fix1/
b#s/word2/fix2/
b#s/word3/fix3/
 .......
b#s/word40/fix40/
```

The first two commands specify that explicit delimiters are to be used and that search errors are to be suppressed. Since explicit delimiters are used, the <ESC> character is not needed anywhere.

Macros do not need to end in <ESC> <ESC>. Search errors must be suppressed, because otherwise, if any word is not found the entire macro will abort.

The second macro reads in each of the 20 files, executes the first macro, writes the file back to disk, and continues with the next file. It is assumed that the first macro is in register 1.

```
EB file1.txt
M1
EY
EB file2.txt
M1
EY
 ........
EB file20.txt
M1
EY
```

Assuming that this macro is in register 0, the following command would invoke the macro to perform the search and replace on all files:

```
M0
```

It is often desirable to save such complex macros on disk for future use. The commands to save these two macros are:

```
RS1 macro1.exc
RS0 macro2.exc
```

Similarly, the commands to retrieve them from disk are:

```
RL1 macro1.exc
RL0 macro2.exc
```

The commands to display them on the console are:

```
RT1          RT0
```

Macros are most easily created in Visual Mode and then moved to the appropriate text register. They can be edited by appending the text register to the end of the current text buffer in visual mode, making the changes and moving them back to the register.

When command macros execute, the contents of a text register are being used. If the macro attempts to change the contents of its own text register or of a register which invoked the macro, unpredictable results could occur. VEDIT checks for this possibility and if it occurs give the error message: "MACRO ERROR". It indicates that you probably made a mistake in using command macros.

Additional Command Macro Features

The "RI" command allows text to be inserted directly into a register. The syntax is "RIrtext", where "r" is the register number, and is otherwise identical to the "I" command. The form "RI+rtext" will append "text" onto any text which is already in register "r".

The "RI" command could be used to manually insert text into a text register, although it is generally easier to move the text in from Visual Mode. The main purpose for the "RI" command is for a command macro to insert text, usually another macro, into a text register. For example, instead of setting up eight registers from eight disk files, it may be easier to just load one disk file and then setup the eight registers with eight "RI" commands.

The "R*" command allows comment lines to be placed in command macros. All text following "R*" through the RETURN is ignored.

```
Command:  R* This demonstrates a comment inside a command macro.
```

The "YT" command types a text string on the console. Its syntax is "YTtext" - the same as the "I" command.

```
Command:  YT Print this on the screen from within the command macro.
```

It is most useful in conjunction with the "YL" command (described below) to send page headers, carriage returns and form feeds to the printer. It can also be used to display progress messages during iteration macros.

```
Example:  @YT/Part 1 is done/      Display a message on console
```

The input and output file names can be displayed on the console. "YR" types the input (read) file name and "YW" types the output (write) file name on the console. The carriage return following the file name may be suppressed by preceding the command with a ":". The file names will commonly be re-routed to the printer or text buffer in conjunction with the "YL" and "YI" commands.

```
Example:  YW   Display output file name and a carriage return

Example:  :YW  Display output file name without carriage return
```

Re-routing Console Output

Any Command Mode console output, which normally goes to the screen, can be re-routed to either the listing device (printer) or the main text buffer. Such re-routing is in effect until the next Command Mode prompt or until it is canceled.

The command "YL" will re-route console output to the printer. It is used in print formatting macros such as our supplied PRINT.EXC. The form "OYL" will stop the re-routing and allow normal console output. Note that the print command "EO" should not be used in conjunction with "YL" (use "T" instead"), because all output following the EO will go back to the console (i.e., "EO" stops the re-routing).

Example: YL ED B: Print the directory of drive B

"YI" re-routes console output to the text buffer. Each character will be inserted at the edit pointer, and the edit pointer incremented. The form "OYI" stops the re-routing. A simple example to try is:

Command: YI EV

This inserts the VEDIT version number into the text buffer. A more elaborate example is given below under numerical capabilities. Inserting text at the end of the buffer with the "YI" command will proceed very quickly. However, inserting text at the beginning of a large file may take on the order of 1/2 second per character.

The command "YI ED*.ASM" will insert all file names in the disk directory, with a file extension of "ASM" into the text buffer. It is therefore possible for a command macro to determine what files are on the disk and automatically edit those files.

Numerical capability

VEDIT has some limited numerical capability through the use of ten "numeric registers". They are accessed with the "XS", "XA" and "XT" commands. The main purpose for these registers is counting.

nXSr	sets numeric register "r" to the value "n"
nXAr	adds the positive or negative value "n" to register "r"
XTr	types the value of register "r" in decimal, carriage returns are automatically supplied at the end of each line. (the form :XTr suppresses carriage returns.

A numeric register is used in the print formatter macro "PRINT.EXC" to count page numbers. A register could also be used to count line numbers. For example, the following command inserts 200 lines of the form "This is line number nnnn", where nnnn increments for each line:

Example: YI 1XS1 200[@I/This is line number / XT1 1XA1]

Print Formatter Command Macro

Your distribution disk contains a command macro PRINT.EXC, which performs simple print formatting. This macro can be used as an alternative to the "EO" print command, which performs no formatting. PRINT.EXC will skip over page perforations and print the file name and page number at the top of each page. A form feed is issued after the text is printed. Much more sophisticated formatters can be written with VEDIT macros; PRINT.EXC is intended as a macro example which is relatively easy to understand and expand. Possible enhancements include line numbering, header and footer messages and more.

To get ready to use PRINT.EXC issue the command:

Command: RL9 PRINT.EXC This loads the print formatter into VEDIT

This loads the command macro into text register 9 where it will remain unless you insert other text into the register.

To print your text, simply issue the command:

Command: M9 This prints the entire text buffer

The entire text buffer should begin printing. You can stop the printing by typing <CTRL-C>.

Feel free to modify the print macro to suit your needs. Modifying this macro is also an informative way to learn more about VEDIT macros. If you find that you use PRINT.EXC a lot, you can automatically load it into VEDIT by placing the command "RL9 PRINT.EXC" into the VEDIT.INI file, which is executed each time VEDIT is invoked.

We hope that the simple PRINT.EXC macro inspires you to write your own more sophisticated macros. The Command Mode is a capable text oriented programming language and the examples in this manual barely scratch the surface of the applications which are possible. The other macro on your disk, ZIL-INT.EXC, is an example of a large brute force search and replace macro, which can perform a translation, in this case from ZILOG Z80 to INTEL style Z80 instructions.

As examples of more complex command macros, we have the optional VHELP and VMAIL disks. VHELP is a menu driven VEDIT tutorial program. The text can be easily modified to act as a tutorial for any other product or procedure you commonly use or need to teach to others. VMAIL is a menu driven mail merge program (command macro) which can print form letters, envelopes, edit the form letter, create and edit the address data base and more. These are available as inexpensive options.

'n' represents a positive number. (# represents 32767)
'm' represents a number which may be negative to indicate backwards in the text buffer.
'r' represents a digit "0 - 9" specifying a text register.

'string', 's1', 's2' and 'text' represent text strings which may include the RETURN key in them. May use explicit delimiters, or else must end in <ESC>.

'file' is a disk file name in normal CP/M (MSDOS) format with optional disk drive, extension and CP/M User Number. Any leading spaces are ignored. Must be ended with a RETURN or an <ESC>.

nA	Append 'n' lines from the input file to the text buffer. "0A" reads as much as possible.
-nA	Read back 'n' lines from the Output file. "-0A" reads back until the text buffer is nearly full.
B	Move the edit pointer to the beginning of the text buffer. "-B" moves to the beginning of the file.
mC	Move the edit pointer by 'm' positions.
mD	Delete 'm' characters from the text.
E	First letter of extended two letter commands.
nFstring<ESC>	Search for the 'n'th occurrence of 'string' in the current text buffer and position the edit pointer after it. 'string' may be up to 32 characters long.
Gr	Insert the text register 'r' at the edit pointer.
H	Display the VHELP.TXT file and load it into memory if necessary. -H deletes it from memory.
Itext<ESC>	Insert the 'text' into the text buffer at the edit pointer. The edit pointer is moved past 'text'.
mK	Kill 'm' lines.
mL	Move the edit pointer by 'm' lines and leave at the beginning of that line.
Mr	Execute text register 'r' as a command macro.
nNstring<ESC>	Search for the 'n'th occurrence of 'string' and perform auto-disk buffering to read more of the file from disk if necessary. The edit pointer is positioned after last 'string' if found.

mPr	Put 'm' lines of text into text register 'r'. 'r' may be preceded by "+" to append to the register. "0Pr" empties text register 'r'.
Ss1<ESC>s2<ESC>	Search for the next occurrence of 's1', and if found, change to 's2'.
mT	Type the next 'm' lines on the console.
U	Print # of free bytes remaining / # bytes in text buffer / # bytes in combined text registers.
V	Go into visual mode. Set cursor position from current edit pointer.
nW	Write 'n' lines to the disk from the beginning of the text buffer and delete from the text buffer. "0W" writes out the text buffer up to the current line.
-0W	Write all lines from the edit pointer to end of text buffer to the "VEDIT.REV" file. This makes more memory space free.
Z	Move the edit pointer past the last character in the text buffer. "_Z" moves to the end of the file.

SPECIAL CHARACTERS

<CTRL-Q>	Literal character. Next character, usually a control character, is taken literally and not interpreted. Allows searching and inserting of control characters including line editing characters, <CR> and <ESC>.
_	Immediately precedes "F" or "S" to cause a global search/replace to the end of the file, instead of just the end of the text buffer. "_F" is equivalent to "N". Precedes "B", "L" or "Z" to cause a global operation.
@	Immediately precedes "F", "N", "S", "I", "RI" or "YT" to indicate that explicit terminating characters are being used.
:	Immediately precedes "F", "N" or "S" to indicate that search error messages are to be suppressed. Immediately precedes "XT" and "YR" and "YW" to suppress <CR><LF> being sent to the console.
#	Represents the maximum positive number 32767. It is used to signify "forever" or "all occurrences of".

EXTENDED COMMANDS

EA	Saves the file being edited on disk and allows further editing to continue.
EBfile	Open the file 'file' for both Read and Write and read the input file. If the file does not exist, "NEW FILE" is printed.
EC	Allow user to change disks. Used for write error recovery, or just to edit files on other disks.
ED	Displays disk directory. Drive specification, "?" and "*" wildcard characters may be used.
EF	Close the current output file.
EGfile[line range]	Insert the line range of the file 'file' into the text buffer at the edit pointer. If no line range is specified, the entire file is inserted.
nEI	Insert the character with decimal value 'n' into the text buffer at the edit pointer. The value "26" is not allowed. Values of 128 to 254 are allowed.
EKfile	Erase (kill) the file 'file' from the disk. This makes more space free on the disk.
ELfile[line range]	Display with line number the line range of the file 'file'. Same syntax as EG command.
mEO	Print (on line printer) the next 'm' lines. "0EO" prints from the beginning of the text buffer to the current line.
EP n k	Change the value of parameter 'n' to 'k'. Currently there are the following parameters:

1	Cursor type (Mem Mapped Only)	(0, 1 or 2)
2	Cursor blink rate (Mem Mapped Only)	(5 - 100)
3	Indent increment	(1 - 20)
4	Lower case convert	(0, 1, 2 or 3)
5	Conditional convert character	(32 - 126)
6	Display line and column position (0=none, 1=line, 2=column, 3=both)	(0 - 3)
7	Word Wrap column (0 = Off)	(0 - 255)
8	Bit 8 Allowed (1=Input, 2=Output, 3=Input & Output)	(0 - 7)
9	Cursor positioning mode	(0 - 2)
10	Virtual line length with scrolling	(40 - 255)
11	Horizontal scrolling increment	(1 - 78)

EQ — Quit the edit session and leave disk files exactly as before the session started. Return to OP system.

ERfile — Open the file 'file' for input. Gives error if file does not exist.

ES n k — Change the value of switch 'n' to 'k'. Currently there are the following switches:

1	Expand Tab with spaces	(0=NO 1=YES)
2	Auto buffering in visual mode	(0=NO 1=YES 2=BACK)
3	Start in visual mode	(0=NO 1=YES)
4	Point past text reg. insert	(0=NO 1=YES)
5	Ignore UC/LC search distinction	(0=NO 1=YES)
6	Clear screen on Visual Exit	(0=NO 1=YES)
7	Reverse Upper and Lower case	(0=NO 1=YES)
8	Suppress search errors	(0=NO 1=YES)
9	Explicit string delimiters	(0=NO 1=YES)

ET n1,n2,n3 ... — Set new tab positions. The ET is followed by up to 30 decimal numbers specifying the tab positions. Alternately, if ET is followed by just one 'n', tabs are set to every 'n' positions.

EV — Display the VEDIT version number.

EWfile — Open the file 'file' for output. Any existing file by that name will be renamed to "file.BAK" following an EF or EX. Gives error if an output file is already open.

EX — Exit back to CP/M after writing the text and any unappended part of the input file to the output file. Gives error if no output file is open.

EY — Finishes editing a file by writing the entire text buffer and any remaining portion of the Input file to the Output file and closing it. Usually followed by an "EB" command.

EZ — exit the edit session and leave disk files exactly as before the session started. Stay in VEDIT.

TEXT REGISTER COMMANDS

RDr	Dump contents of register 'r' on console. Control characters are not expanded.
RIr text<ESC>	Insert the text 'text' into text register 'r'. Use "RI+r" to append to any existing contents.
RLrfile	Load register 'r' from file 'file'.
RPr	Send contents of register 'r' to line printer.
RSrfile	Save contents of register 'r' in file 'file'.
RTr	Type contents of register 'r' on console. Control characters are expanded, <ESC> is represented as "$". An encountered <CTRL-S> "stop character" causes a pause - type any character to continue.
RU	Display number of characters (size) in each text register.
R*	Treat all following characters up to RETURN as a comment.

NUMERIC REGISTER COMMANDS

nXAr	Add 'n' to numeric register 'r'.
-nXAr	Subtract 'n' from numeric register 'r'.
nXSr	Set numeric register 'r' to 'n'.
XTr	Type numeric register 'r' in decimal. Use ":XTr" to suppress <CR<LF>.

MISCELLANEOUS COMMANDS

YI	Re-route console output until next command prompt to text buffer at edit pointer and update edit pointer. (May work very slowly) Use "-YI" to disable "YI".
YL	Re-route console output until next command prompt to list device (printer). Use "-YL" to disable "YL". Note: YL is disabled by "E0" command.
YR	Type the input (read) file name on the console. Use ":YR" to suppress the <CR><LF> following the file name.
mYS	Strip the 8th bit from all characters in the specified line range.
YTtext<ESC>	Type the 'text' on the console.
YW	Type the output (write) file name on the console. Use ":YW" to suppress the <CR><LF> following the file name.

PATTERN MATCHING CODES

- ¦X Matches any character (equivalent to use of "¦" alone in previous versions of VEDIT)
- ¦A Matches any alphabetic letter, upper or lower case
- ¦B Matches a blank - a space or a tab
- ¦C Matches any control character - a character with a value of 0 - 31 (decimal)
- ¦D Matches any numeric digit - "0" through "9"
- ¦L Matches any line terminator - Line Feed, Form Feed or the End Of File
- ¦M Matches multiple characters - zero, one or more characters so that the string following the "¦M" is satisfied
- ¦N Matches any character which is NOT the following single character or pattern matching code
- ¦R Matches any alphanumeric character - a letter or a digit
- ¦S Matches any separator - a character which is NOT a letter or digit
- ¦U Matches any upper case letter
- ¦V Matches any lower case letter
- ¦W Matches "white space" - one or multiple spaces and/or tabs
- ¦¦ Matches a "¦" - this is the literal "¦".

nA Append

Example: 100A 0A -0A

Description: This command will append 'n' lines from the input file to the end of the text buffer. Fewer lines will be appended if there is insufficient memory space for 'n' lines, or there are not 'n' lines remaining in the input file. If 'n' is 0, an auto-read is performed, which reads all of the input file or until the main memory is almost full. The command can be issued (with 'n' not zero) after an auto-read to read in more of the file. The command is ignored if there is no input file open. The input file can be opened with the "EB" and "ER" commands, or when VEDIT is invoked.

The special forms "-nA" and "-0A" will read back 'n' lines from the Output file into the beginning of the text buffer. "-0A" reads all of the Output file back or until the text buffer is almost full. Nothing is read back if there is no Output file or it is empty.

Notes: No indication is given if fewer than 'n' lines were appended. Use the "U" command to see if anything was appended. If the text buffer is completely full, the text registers cannot be used and Visual Mode will not work well.

See Also: Commands: U, W, EB, EG, ER
Automatic Disk Buffering

Examples:

ERTEXT.DOC

0A	The file 'TEXT.DOC' is opened and all of the file is read in, or until the memory is almost full.
-0A	Read as much of the Output file as will fit back into the beginning of the text buffer.

B Beginning

Example: B _B

Description: This command moves the edit pointer to the beginning of the text buffer. The beginning of the text buffer will not be the beginning of the text file if a "W" command or an auto-write was done. In this case, use the command "_B" to move back to the beginning of the text file.

Notes:

See Also: Commands: EA, Z
Backward Disk Buffering

Examples: B12T Moves the edit pointer to the beginning of the text buffer and types the first 12 lines.

mC Change

Example: 12C -4C

Description: This command moves the edit pointer by 'm' character positions, forwards if 'm' is positive and backwards if 'm' is negative. The edit pointer cannot be moved beyond the beginning or the end of the text buffer, and an attempt to do so will leave the edit pointer at the beginning or the end respectively. Remember that every line normally ends in a <CR> <LF> (carriage return, line feed), which represents two character positions.

Notes:

See Also: Commands: D, L

Examples: Fhello$-5C$$ Searches for the word "hello", and if it is found, positions the edit pointer at the beginning of the word.

mD Delete

Example: 12D -4D

Description: This command deletes 'm' characters from the text buffer, starting at the current edit pointer. If 'm' is positive, the 'm' characters immediately at and following the edit pointer are deleted. If 'm' is negative, the 'm' characters preceding the edit pointer are deleted. Fewer than 'm' characters will be deleted if the ends of the text buffer are reached.

Notes:

See Also: Commands: C, K

Examples:

100[FBIKES$-D$]	The 'S' will be deleted from up to 100 occurrences of the word 'BIKES'.

E Extended Commands

Example: EX EV

Description: This is not a command by itself but just the first letter of all the extended commands.

Notes: No error is given if just E$$ is given.

See Also: Extended commands.

Examples:

nF string<ESC> Find

Example: Fmispell$$ 10Fwords$$ F$$ @F/¦Sword¦S/

Description: This command searches the text buffer, beginning from the edit pointer, for the 'n'th occurrence of 'string'. The edit pointer will be positioned after 'n'th occurrence of 'string' if it is found. If the 'n'th occurrence of 'string' is not found, an error will be given (unless suppressed) and the edit pointer will be positioned after the last occurrence of 'string' found, or be left at its original position if no occurrences were found. If no string is specified, the search will reuse the previously specified string. The switch "Ignore Upper/Lower case distinction" will determine if the search will ignore the distinction between upper and lower case letters. The form "n_Fstring<ESC>" will perform a global search to the end of the file for "string" instead of just to the end of the text buffer.

Notes: The search is always forward, never backwards. Remember that Pattern Matching Codes can be used. The "@" option allows an explicit delimiting character. For the command form "#Fstring<ESC>", an error is only given if no occurrences of 'string' are found.

See Also: Command: N

Examples:

BFhello$$	Searches for the word "hello" from the beginning of the text buffer.
#[3Ffirst$-5DIthird$]	Changes every third occurrence of the word "first" to "third".
Z-100LFend$$	Find the word "end" if it occurs in the last 100 lines of the text buffer.
#[@F/fix up/V]	Finds the next occurrence of the string "fix up" and enters Visual Mode. Any changes can be made in Visual mode. When Visual Mode is exited, the next occurrence of "fix up" is found and so on.
F$V	The next occurrence of the previous specified string is found, and Visual Mode is then entered.

Gr Get

Example: G4

Description: This command inserts a copy of text register 'r' at the current edit pointer. If there is insufficient memory space for the entire copy, nothing is inserted and an error message is given. If the text register is empty, nothing is inserted. The contents of the text register are not affected by this command.

Notes: The "P", "RL", "RI" commands or Visual Mode are used to place text in a text register.

See Also: Commands: P
Visual Mode Text Registers

Examples:

BG9	Inserts the contents of text register 9 at the beginning of the text buffer.
12[G2]	Inserts the contents of text register 2 twelve times at the current edit pointer.
132P3$132K _B 10LG3	Moves 132 lines of text, by saving it in text register 3, killing the original lines and inserting the text after the tenth line of the file, in the situation where the beginning of the file is no longer in the text buffer.

H Help

Example: H -H

Description: This command loads a special help file called VHELP.TXT into memory and displays the contents one screen at a time (you press any key to see the next screen, or <CONTROL-C> to return to command mode). Since the help file does take up memory space, it can be eliminated after use by typing -H. Alternatively, you can leave it in memory for a quicker access next time you type the H command.

See Also: Command Mode
Memory and File Management

I text<ESC> Insert

Example: Ia word$$ I<CR>new line$$

Description: This command inserts the text 'text' into the text buffer, starting at the current edit pointer. The insertion is complete when the <ESC> (or explicit delimiter) character is encountered. The inserted text does not overwrite any existing text. The 'text' may contain the RETURN key, which is expanded to carriage return - line feed. If insufficient memory space exists for the 'text', an error will be given and only part of the 'text' will have been inserted. The edit pointer is moved just past the inserted text. This command is probably best used in iteration macros, since normal text insertion is much easier to do in Visual Mode.

Notes: Control characters including <ESC> can be inserted by preceding them with the literal character <CTRL-Q>. The "@" character allows an explicit delimiting character to be used. The tab character is not expanded with spaces as is optional in Visual Mode.

See Also: Commands: EI

Examples:

200[I<CR><TAB>$] Inserts 200 new lines, each beginning with a tab character.

Iunder<CTRL-Q><CTRL-H> $$ Inserts the text "under", a BACKSPACE and the underline character. This will underline the "r" on some printers.

@I/a word/ Inserts the text "a word" into the text buffer.

@I/EP 7 70<ESC><CR>/ Inserts the command line "EP 7 70 <ESC>" into the text, including a RETURN.

mK Kill

Example: 4K -3K OK #K

Description: This command kills (deletes) the specified number of lines. If 'm' is positive, all text from the current edit pointer up to and including the 'm'th <LF> is deleted. If 'm' is negative, all text preceding the edit pointer on the current line and the 'm' preceding lines are deleted. If 'm' is 0, all characters preceding the edit pointer on the current line are deleted. Fewer than 'm' lines will be killed if either end of the text buffer is reached.

Notes:

See Also: Command: D, T

Examples:

#[Ftemp line$OLK]	Kills all lines which contain the string "temp line".
-#K	Kills all text before the edit pointer.
#P6#K	Saves the rest of the text from the edit pointer in text register 6 and then deletes it from the text buffer.

mL Lines

Example: 120L -14L 0L 1000_L

Description: This command moves the edit pointer by 'm' lines. If 'm' is positive, the edit pointer is moved to the beginning of the 'm'th following line. If 'm' is negative, the edit pointer is moved to the beginning of the 'm'th preceding line. If 'm' is 0, the edit pointer it moved to the beginning of the current line. Attempting to move past either end of the text buffer will leave the edit pointer at the respective end. The global edit command "m_L" will move past the end of the text buffer, with automatic disk buffering if necessary.

Notes: If an "L" command in an iteration macro attempts to go past the end of the text buffer it will end the current iteration level.

See Also: Commands: C, T

Examples: #[Stypo$type$0LT] Changes all occurrences of "typo" to "type" and types out every line that was changed.

Mr Macro

Example: M1

Description: This command executes the contents of register 'r' as a command macro. Any legitimate command or sequence of commands may be executed as a macro. Macros are most easily created and edited in Visual Mode. They may also be saved and loaded from disk or inserted with the "RI" command. A macro may invoke another text register, which in turn may invoke another, up to a nesting depth of 5. Macros are very convenient for holding long command sequences which are repeatedly used, saving the effort of retyping them each time.

Notes: RETURNs may be used to separate commands in order to improve readability. The error "MACRO ERROR" results if a macro attempts to change a text register which contains the executing command macro. An "M" without a following digit is interpreted as "M0".

See Also: Commands: G, P, RI, RL, RS, R*
Visual Mode Text Registers, Command Macros.

Examples: See heading "Text Register Macros" for an example.

nNs1<ESC> Next

Example: Nbad line$$ 3@N/third/ N$$

Description: This command is very similar to the "F" command, except that if the 'n'th occurrence of 's1' is not found in the text buffer, forward disk buffering is performed to read in more of the input file until the 'n'th occurrence is found or the end of the input file is reached. If the 'n'th occurrence still is not found, an error is given. The edit pointer is positioned very similarly to the "F" command. However, with the "N" command it is possible that the 'n'th occurrence is not found, and that the previous occurrence is no longer in the text buffer due to auto-buffering. In this case the edit pointer is positioned at the beginning of the text buffer.

Notes: All Notes for the "F" command also apply. "_F" is identical to "N" and "_F" should be used, since "N" may be discontinued in a future release. The error "NO OUTPUT FILE" occurs if no output file is open for performing forward disk buffering.

See Also: Command: F, _F Auto Buffering

Examples: #[Ntypo$-4DItype$] Changes all occurrences of the string "typo" to "type" in the rest of the file.

#[@N/typo/-4D@I/type/] Alternate form of the same command using explicit delimiters.

mPr Put

Example: 40P1 -20P+2 0P3

Description: This command saves a copy of the specified text lines in text register 'r'. The previous contents of the text register are destroyed, unless the 'r' is preceded with a "+" indicating that the text is to be appended. The range of lines saved is the same as for the "K" or "T" commands. If 'm' is zero, the text register is simply emptied, and nothing is saved in it. Since the text buffer and the text registers share the same memory space, saving text in the text registers decreases the amount of memory available to the text buffer. Thus the "0Pr" command should be given when the text in a register is no longer needed. This command does not change the text buffer. If there is insufficient memory space for the text copy, the text register is only emptied, nothing is saved in it and an error is given. The saved text is inserted in the text buffer with the "G" command or in Visual Mode.

Notes: The error "MACRO ERROR" results if a macro attempts to change a text register which contains the executing command macro.

See Also: Commands: G, K, T
Visual Mode text move

Examples:

120P1$120K	The text lines are saved in text register 1 and are then deleted from the text buffer.
-23T -23P6	The text lines are typed for verification before they are saved in the text register.

nSs1<ESC>s2<ESC> Substitute

Example: Stypo$type$$ #Sname$Mr. Smith$$ _Sold$new$$

Description: This command performs 'n' search and substitute (replace) operations. Each operation consists of searching for the next occurrence of 's1' in the text buffer and changing it to 's2'. An error is given if 's1' is not found. If there is insufficient memory space for inserting 's2', 's1' will have been changed to as much of 's2' as possible and an error is given. The edit pointer is positioned after 's2', if 's1' is found, or else is left at its original position if 's1' is not found. For the command form "#Ss1<ESC>s2<ESC>, an error is only given if no occurrences of 's1' are found. The form "n_Ss1<ESC>s2<ESC" will perform a global search and replace, searching to the end of the file if necessary.

Notes: All Notes for the "F" command apply here too. A command like #Sfishes$fish$$ will execute much faster than the equivalent command #[Sfishes$fish$].

See Also: Commands: F, N, I

Examples:

#Stypo$type$$ Changes all occurrences of "typo" to "type".

#[Stypo$type$OLT] Changes all occurrences of "typo" to "type" and types out every line that was changed.

ES 9 1
#[S/typo/type/OLT] Alternate form of above command. Explicit delimiters can now be used without "@" prefix.

#[Sname$smith$V] Change the next occurrence of "name" to "smith" and enter into Visual Mode. Any changes can be made in Visual Mode and when Visual Mode is exited, the next occurrence of "name" will be searched and so on.

#_Sgarbage$$ Deletes all occurrences of the string "garbage" from the rest of the 1file0.

mT Type

Example: 14T -6T 0T

Description: This command types out (displays) the specified lines. If 'm' is positive, all characters from the edit pointer up to and including the 'm'th <LF> are typed. If 'm' is negative, the previous 'm' lines and all characters up to just preceding the edit pointer are typed out. If 'm' is 0, only the characters on the present line preceding the edit pointer are typed out. Fewer than 'm' lines will be typed out if either end of the text buffer is reached. Note that "0TT" will display the current line regardless of the position of the edit pointer on it. This command does not move the edit pointer. This command is most useful in iteration macros for displaying selected lines. Visual Mode should be used for looking at sections of a file.

Notes:

See Also:

Examples: #[Fmoney$0TT] Types out every line in the text buffer with the string "money" in it.

U Unused (Free Memory)

Example: U

Description: This command displays the number of memory bytes free for use by the text buffer or text register, followed by the number of memory bytes used by the text buffer (length of the text buffer), followed by the combined number of memory bytes used by the text registers (length of the text registers).

Notes: These three numbers will not always add up to the same total, since several other small buffers all use the same memory space. If the number of free bytes goes below 260, the "FULL" flag will be set when in Visual Mode.

See Also:

Examples:

V Visual

Example: V

Description: This command enters Visual Mode. The visual cursor position will be set from the current edit pointer position. Visual mode is exited with either the [VISUAL EXIT] or the [VISUAL ESCAPE] key. At that time the edit pointer will be set from the cursor position.

Notes: The text registers are preserved.

See Also: Visual Mode

Examples: Fhere$V$$ Finds "here" and enters Visual Mode.

nW Write

Example: 20W #W 0W -100W

Description: This command writes 'n' lines from the beginning of the text buffer to the output file and then deletes these lines from the text buffer. If there are less than 'n' lines in the text buffer, the entire text buffer is written out. If 'n' is zero, the entire text buffer up to the line the edit pointer is on, is written out. If no output file is open, an error is given and no text is written. The output file can be opened with an "EW" or "EB" command or when VEDIT is invoked.

The special forms "-nW" and "-0W" write the end of the text buffer to the temporary ".R" file. These commands are primarily used to make more memory space available for further edit operations. "_nW" writes the last 'n' lines in the text buffer out to disk. "_0W" writes out the end of the text buffer beginning with the line the cursor is on.

Notes: No indication is given if less than 'n' lines were written.

See Also: Commands: A, EB, EW, EX

Examples:

```
EW part1.txt
24W
EF
EW part2.txt
EX
```

The first 24 lines of the text buffer are written out to file "PART1.TXT" and the rest of the text buffer is written out to file "PART".TXT" and the edit session is completed.

Z Zip

Example: Z _Z

Description: This command moves the edit pointer to the last character in the text buffer. The command "_Z" moves the edit pointer to the last character in the file, if necessary performing forward disk buffering.

Notes:

See Also: Commands: B, N
Disk Buffering

Example:

Z-100L	Positions the edit pointer to the 100th line before the end of the text buffer.
Z-12T	Types out the last twelve lines in the text buffer.
_Z-12T	Types out the last twelve lines in the file.

EA Edit Again

Example: EA

Description: This command writes the entire text buffer and any remaining part of the input file out to the output file and closes the output file. All file backup and renaming is performed as with the "EY" or "EX" command. The output file is then reopened as both the input and output file and the input file is read back in. Finally, it restores your original position in the file and any text markers. This command thus starts a new edit session and is functionally similar to an "EX" command followed by invoking VEDIT again with the name of the current output file. The main purpose of this command is to save the currently edited file on disk as a safeguard against losing the file due to a user error, or hardware, software or power failure. The contents of the text registers are not affected by the "EA" command.

Notes:

See Also: Commands: B, G, EX, EY
Visual Restart

EA Z V	Saves the current file on disk, moves the edit pointer to the end of the text buffer and re-enters Visual Mode. Useful for periodic saving of ongoing work.

EBfile<ESC> Edit Backup

Example: EBfile.txt

Description: This command opens the file 'file' for both input and output and then reads in all of the file, or until the memory is nearly full. If an output file is still open, an error is given and the command has no other effect. "EB file" is similar to invoking VEDIT with the command:

VEDIT file

It is also similar to the sequence of commands:

ERfile<ESC>EWfile<ESC>0A

except that if the file does not yet exist on disk, the message "NEW FILE" is displayed.

Notes: The term "backup" is used here to describe this command since the term is used by some other editors to perform a similar operation. Remember that VEDIT always creates a "backup" of a file on disk, if its name is used as the name of the output file.

See Also: Commands: W, ER, EW

Example:

```
EY
EB newfile.txt
```

The entire text buffer is written out to the current output file, that file is closed, and the file "NEWFILE.TXT" is opened for input and output and read in.

```
ER part1.txt$0A
EB part2.txt
```

The file "PART1.TXT" is read into the text buffer, the file "PART2.TXT" is then made the current input and output file and is appended to the end of the previous file "PART1.TXT".

EC Edit Change (Disk)

Example: EC

Description: This command must be given before you attempt to change any logged-in disks in order to recovery from a disk write error, or to read files from another disk. An error is given if the current disk still has an output file open. This command is used in the event of a disk write error where you do not wish to delete any files with the "ED" command. In this case the "EF" command should be given to close that part of the output file which has been written to the original disk. Then issue the "EC" command. It will prompt with a message when the original disk can be removed and a new disk inserted. Type a [RETURN] after the new disk is inserted and then issue an "EW" command to open a file for output. You can then issue any "W" command or the "EX" command. When the edit session is over the output file is in two parts on two disks. They can easily be merged with a CP/M PIP (MSDOS COPY) command or with VEDIT. See the "ER" command for this. This command can also be used to switch to another disk before an "ER" or "EG" command.

Notes: Be sure that the entire input file has been read into memory before issuing the "EC" command.

See Also: Commands: ED, EF
Disk Write Error Recovery.

Example: EC — Will give prompt: INSERT NEW DISK AND TYPE RETURN when the user should remove the old disk and insert a new disk.

ED Disk Directory

Example: ED EDA: ED B:*.TXT

Description: This command displays the disk directory of any drive. It serves as a reminder of the names of files you wish to edit, merge or have already written out. No files or text is changed. Drive specifiers, the "?" and "*" wildcard characters and CP/M User Numbers are allowed.

Notes:

See Also: Disk Write Error Recovery

Example:

ED B:*.TXT	Gives the directory of all files with extension ".TXT" on drive B.
ED B:=3	Gives the directory of all files CP/M drive B, with CP/M User Number 3.

EF Edit Finish (Close)

Example: EF

Description: This command closes the output file, making the text which has ALREADY been written to disk permanent. Any existing file on disk with the same name as the output file is backed-up by renaming it with a file extension of ".BAK".

Notes: This command DOES NOT actually write any text to disk. Use the "EY" command to write the text to disk and save it. Since the output file is actually opened with the file extension ".$$$", the ".$$$" file is first closed, then any existing file on disk with the same name as the output file is renamed to ".BAK", and last, the ".$$$" file is renamed to the true output file name. (See EW command notes.)

See Also: Commands: EW, EX, EY

Example:

EW save.txt

100W EF	The first 100 lines of the text buffer are written out as the file "SAVE.TXT".

EGfile[line range] Edit Get (File)

Example: EGfile.txt[1,100] EG file.txt

Description: This command will insert a specified line number range of the file "file" into the text buffer at the edit pointer. If insufficient memory exists to insert the entire file segment, as much as possible will be inserted and a *BREAK* message will be given. If no line range is specified, the entire file is inserted.

Notes: The line numbers of a file can be displayed using the "EL" command. A space may be used instead of a comma in the "line range".

See Also: Commands: A, EL, ER

Example: EG library.asm[34 65] Lines 34 through 65 of the file "LIBRARY.ASM" are inserted into the text buffer at the edit pointer.

nEI Edit Insert

Example: 12EI

Description: This command will insert the character whose decimal value is "n" into the text buffer at the edit pointer. This is useful for entering special control characters into the text buffer, especially characters which cannot be generated from the keyboard. Characters with a decimal value between 128 and 255 can also be entered with the EI command. Only the "End of File" marker with a value of 26 cannot be entered. Control characters are displayed in both command and Visual Mode by preceding the letter with a "caret" (^).

Notes:

See Also: Commands: I, [NEXT CHAR LITERAL]

Example: 8EI A backspace character is inserted into the text buffer at the edit pointer.

92EI A "\" is inserted into the text with the EI command, because it could not be generated from the keyboard.

EKfile<ESC> Edit Kill

Example: EKfile.txt EK*.bak

Description: This command will erase (kill) the file 'file' from the disk. This is the easiest method of recovering from a disk write error in order to make more disk space or a free entry in the directory. The command first displays a directory of the files to be erased and asks for confirmation before erasing them.

Notes: Never erase any ".$$$" or ".R" files from within VEDIT! These are the temporary files VEDIT is using. Don't delete the input file until all of it has been read into memory.

See Also: Commands: EC, ED
Disk Write Error Recovery

Example:

EK oldfile.txt	The file "OLDFILE.TXT" is erased from the disk.
EK *.bak	Deletes all files with a file extension of ".BAK" from the default drive.

ELfile[line range] Edit Look (File)

Example: EL file.txt EL b:file.txt[200,400]

Description: This command allows you to view another file with line numbers, in the specified line number range. If no line range is specified, the entire file is displayed. The displayed line numbers may subsequently be used with the "EG" command to extract a portion of the file.

Notes:

See Also: Commands: EG

Example:

EL library.asm	The file "LIBRARY.ASM" is displayed with line numbers.

mEO Output to Printer

Example: 40EO -20EO 0EO

Description: This command sends the specified lines to the listing device (printer). If 'm' is positive, all text from the edit pointer up to and including the 'm'th line is printed. If 'm' is negative, the previous 'm' lines and all characters up to the edit pointer are printed. If 'm' is 0, the entire text from the beginning of the text buffer up to the current edit pointer is printed. Fewer than 'm' lines will be printed if either end of the text buffer is reached. This command does not move the edit pointer.

Notes: The print out can be stopped by typing <CTRL-C>.

See Also: Commands: L, T, RP, YL
Printing Text from Visual Mode

Example: ZOEO Prints the entire text buffer and positions the edit pointer at text end.

EP n k<ESC> Edit Parameters

Example: EP 1 4 EP 3 30

Description: This command changes the value of parameter 'n' to 'k'. The numbers are specified in decimal and separated by spaces or commas. The default values of these parameters are determined during the customization process. An error is given if 'n' is specified out of range. The parameters are:

1	Cursor type	(0, 1 or 2)
2	Cursor blink rate	(5 - 100)
3	Indent Increment	(1 - 20)
4	Lower case convert	(0, 1, 2 or 3)
5	Conditional convert character	(32 - 126)
6	Display line and column number	(0, 1, 2 or 3)
7	Word Wrap column	(0 - 255)
8	Bit 8 Allowed	(0 - 7)
9	Cursor positioning mode	(0 - 2)
10	Horizontal scrolling margin	(40 - 255)
11	Horizontal scrolling increment	(1 - 78)

Parameter (1) determines the type of cursor displayed in Visual Mode for memory mapped versions. The CRT terminal versions use the terminal's cursor instead. The cursor types are: 0=Underline, 1=Blinking Reverse Video Block, 2=Solid Reverse Video Block, 3=Attribute (SSM VB3 and IBM PC only)

Parameter (2) determines the cursor's blink rate for cursor types 0 and 1 above.

Parameter (3) determines how much further the editor will indent each time the [INDENT] key is typed. The indent position after typing the [INDENT] key four times is therefore the "Indent Increment" multiplied by four.

Parameter (4) determines whether lower case characters are converted to upper case. For value (0) no conversion takes place, for (1) all lower case are converted to upper case. For (2) lower case are converted to upper case, unless the cursor is past a "special" character on the text line. This "special" character is set by parameter (5). Mode (3) is similar to (2) except that characters are reversed instead forced to upper case. All of this is primarily applicable to assembly language programming, where it is desirable to have the Label, Opcode and Operand in upper case and the comment in upper and lower case.

Parameter (5) sets the conditional upper/lower case convert character used for parameter (4) above.

Parameter (6) determines whether the cursor's line position in the file and horizontal position on the line are displayed on the status line. The values are: 0 = Both off, 1 = Line number displayed, 2 = column displayed and 3 = both displayed.

Parameter (7) is the Word Wrap column. It is also the right margin used when formatting paragraphs. A value of 0 disables both Word Wrap and formatting. It should be turned off when editing programs!

Parameter (8) determines where 8 bit characters are allowed and is a 3 bit parameter. Setting Bit 1 allows 8 bit input characters. Bit 2 allows 8 bit characters on output (alternately, the 8th bit will be stripped and the character displayed in reverse video if possible). Bit 3 allows unused control sequences to be inserted into the text buffer. Normal value is "1" - allow 8 bits on input, display 8 bit characters in reverse video. Users with the IBM PC and NEC APC, which have alternate character sets will want to use a value of "3".

Parameter (9) determines the cursor positioning mode. The modes are 0 = cursor only at real text, 1 = cursor allowed past end of lines, 2 = pad with spaces when past end of line.

Parameter (10) is the horizontal scrolling margin which sets the maximum right margin with scrolling. Text lines longer than this "scroll margin" are wrapped to the next screen line.

Parameter (11) is the horizontal scrolling increment. It determines how much the screen scrolls right or left when [SCROLL RIGHT] and [SCROLL LEFT] are pressed or VEDIT scrolls automatically.

Notes: The parameter values are specified in decimal.

See Also: Commands: ES, H (to help you remember them all)
Customization, Visual Mode, Indent and Undent Functions

Examples:

EP 1 6	This sets the "Indent Increment" to six.
EP 7 70	This sets the Word Wrap column to 70.

EQ Edit Quit

Example: EQ

Description: This command quits the edit session without saving the current file and leaves VEDIT. Its main purpose is to "quit" after one has finished just looking at a file or one doesn't want to save any of the edit changes made. DO NOT confuse this command with the "EA" command; their results are quite opposite. Remember that the "EA" command saves the file and starts a new edit session.

Notes: Any output file with the file extension ".$$$" will also be deleted. Any original file on disk with the same name as the output file, but with an extension of ".BAK" will have been deleted if any characters were written to the (now deleted) output file. With the exception of this possible backup file, all other files will exist on disk just as they did before the quit edit session.

See Also: Commands: EA, EZ

Example:

#K	Shoot!! Meant -#K
EQ	Since a bad mistake was made in the above command, it is best to quit this edit session, go back to the operating system and start over. All edit changes are lost, but not your file!!

ERfile<ESC> Edit Read

Example: ER newfile.txt

Description: This command opens the file 'file' for input (reading). Nothing is read into the text buffer with this command. The "A" command or an auto-read is used to actually read the input file. If the same file was already open for input, the file is "rewound", so that the file can again be read from the beginning. An error is given if 'file' does not exist. Files can also be read from disks which are not currently running by using the "EC" command. Issue the "EC" command, insert the new disk into a drive which is not being used for any output file and open a file for reading with the "ER" command. This may be necessary in case a file has been split into two parts during a disk write error recovery.

Notes: File names may be preceded with spaces for readability.

See Also: Commands: A, EC, EB, EW

Example:

```
ER parts.inv
20A
```

The file "PARTS.INV" is opened for input and twenty lines from it are appended to the end of the text buffer.

ES n k<ESC> Edit Set

Example: ES 1 0 ES 3 1

Description: This command changes the value of switch 'n' to 'k'. The numbers are specified in decimal and separated by spaces or commas. The default values of these switches are determined during the customization process. An error is given if 'n' is specified out of range. The switches are:

1	Expand Tab with spaces	(0=NO 1=YES)
2	Auto buffering in Visual Mode	(0=NO 1=YES 2=BACK)
3	Start in Visual Mode	(0=NO 1=YES)
4	Point past text reg. insert	(0=NO 1=YES)
5	Ignore UC/LC distinction in search	(0=NO 1=YES)
6	Clear screen on visual exit	(0=NO 1=YES)
7	Reverse Upper and Lower case	(0=NO 1=YES)
8	Suppress search errors	(0=NO 1=YES)
9	Use explicit string delimiters	(0=NO 1=YES)

Switch (1) determines whether or not the tab key in Visual Mode is expanded with spaces to the next tab position. If not, a tab character is inserted into the text buffer. Except for special applications, the tab key should not normally be expanded with spaces.

Switch (2) determines whether auto-buffering is enabled in Visual Mode. "0" disables auto-buffering, "1" enables only forward disk buffering, and "2" enables both forward and backward disk buffering. Unless you have a hard disk we recommend a value of "1". Before using "2", make sure you have sufficient free disk space. Use "0" when you are giving explicit Read/Write commands. This will prevent unexpected disk read and write from occurring while you are editing in Visual Mode.

Switch (3) determines whether or not the edit session will begin in Visual Mode.

Switch (4) determines the edit pointer's position (or cursor's in Visual Mode) following insertion of the text register. If the switch is off, the edit pointer is not moved, and is thus left at the beginning of the newly inserted text. If the switch is on, the edit pointer is moved just past the newly inserted text.

Switch (5) determines whether VEDIT will make a distinction between upper and lower case letters in searches and substitutes using the "F", "N" and "S" commands and during Visual Mode search and replace operations. Most users will probably wish to ignore the distinction, so that the string "why" will match "Why", "WHY" and "why". Setting the switch to "1" will make VEDIT ignore the distinction between upper and lower case characters during searches.

Switch (6) determines whether the screen will be cleared when Visual Mode is exited and Command Mode entered. If the screen is not cleared, the Command Mode prompt "COMMAND:" will appear below the status line. Setting the switch to "1" will clear the screen when Visual Mode is exited.

Switch (7) determines whether all letters typed on the keyboard will be reversed with respect to upper and lower case. It should normally be OFF, but does allow a user with an upper case only keyboard to enter lower case letters. Setting the switch to "1" will make VEDIT reverse all keyboard letters in both Command and Visual Mode.

Switch (8) determines whether search errors will be suppressed. If not suppressed, not finding a string will cause an error message and the command to be aborted. Search errors are usually only suppressed for command macros.

Switch (9) determines whether explicit string delimiters can be used without having to specify the "@" command modifier. This is a matter of personal preference, but is useful with command macros.

Notes:

See Also: Customization, Visual Mode

Example: ES 1 1 This enables tabs typed in Visual Mode to be expanded with spaces.

ET Edit Tab

Example: ET 20 40 60 80 100 120 ET 8

Description: This command changes the tab positions used by VEDIT for displaying tab characters, and in Visual Mode, when the "Expand Tab" switch is set, for expanding tab characters. Up to 30 tab positions are allowed and they must be in the range 1 - 254. The default positions are set during customization. If only one number 'n' is given, the tab positions will be set to every 'n' positions. For word processing the tabs can be set to the same positions as are specified for the print formatting program in order to see how they will look in the final product. An error is displayed if a bad position is given. No tab is needed at position 1, and counting starts at 1 (not at zero). Thus the normal tab positions are:

9 17 25 33 41 49 57 65 73 81 89 97 105 113 121 ...

Notes: For use in Visual Mode, there must be at least one tab position per screen line, i.e. at least one tab every 64 or 80 positions.

See Also: Customization, Visual Mode, Indent and Undent Functions

Example:

EV Edit Version

Example: EV

Description: This command displays the VEDIT version number. This number should be used in any correspondence you have with us concerning the operation of VEDIT. This command can also be used inside iteration macros to give some indication of the progress being made in long executing macros.

Notes:

See Also:

Example:

EWfile<ESC> Edit Write

Example: EW newdat.inv

Description: This command opens the file 'file' for output and subsequent writing. No text is actually written by this command. An output file must be opened in order to save any text on disk. A file can also be opened by the "EB", "EA" commands and when VEDIT is first invoked. If a file is already open for output, an error is given and no other action takes place.

Notes: The file opened is actually a temporary file with the same name, but with an extension of ".$$$". The file is not made permanent and given its true name until it is closed with the "EA", "EF", "EX" or "EY" commands. At that time, any existing file on disk with the same name as the output file is backed up by renaming it with an extension of ".BAK". Any existing file on disk with that name and the .BAK extension will be deleted when any text is written to the output file.

See Also: Commands: W, EA, EF, EX, EY

Example:

```
EW part1.txt
24W
EF
EW part2.txt
EX
```

The first 24 lines of the text buffer are written out to file "PART1.TXT" and the rest of the text buffer is written out to file "PART".TXT" and edit session is completed.

```
ER a:bigfile.asm
EW b:bigfile.asm
OAV
```

Typical procedure for editing a file which is too big for both it and its Backup to fit on the same disk. In this case, it is read from disk A: and written to disk B:. Just be sure that disk B: is nearly empty.

EX Edit Exit

Example: EX

Description: This is the normal way to save the file being edited on disk and exit VEDIT. It writes the entire text buffer to the output file, followed by any unread portion of the input file, closes the output file and exits VEDIT. All file backup and renaming is done as with the "EF" command. The error "NO OUTPUT FILE" is given if no output file is open. The error "NO DISK SPACE" results if there is insufficient disk space to save the entire file.

Notes: In case of a "NO DISK SPACE" or "NO DIR SPACE" error, see the heading "Disk Write Error Recovery" for the procedure to save your file.

See Also: Commands: EA, EB, EF, EQ, EW, EY

Example:

```
VEDIT FILE.TXT
V
EX
```

The editor is invoked in the normal way to edit a file in Visual Mode. The new file is then saved on disk.

EY Finish Edit Session

Example: EY

Description: This command writes out the file being edited to disk and closes it, in preparation for editing another file. First the text buffer is written to the Output file. Any unread portions of the input file at then transferred to the Output file and the Output file closed. All file backup and renaming of files is done as with the "EF" command. This command is equivalent to "EX", but without leaving the editor. The error "NO OUTPUT FILE" is displayed if no output file is open.

Notes: See notes for EF and EX commands.

See Also: Commands: EX, EF

Example:

```
EF
EB newfile.txt
```

The current file is saved on disk, and the file "NEWFILE.TXT" opened for editing.

RDr Register Dump

Example: RD3

Description: This command types (dumps) out the contents of text register 'r' on the console. Control and Tab characters are not expanded. The command is most useful for sending initialization sequences to a CRT terminal, such as sequences to setup programmable function keys. The "RT" command should be used to view the registers, since control characters are then expanded.

Notes: Type CTRL-C to stop the RD command.

See Also: Commands: RT
Auto-Startup

Example: RD5 The contents of text register 5 are dumped (sent) to the console.

RIr text<ESC> Register Insert

Example: @RI3/B#EO/

Description: This command places the following 'text' into text register 'r'. If 'r' is preceded by a '+' the 'text' is appended to any existing contents in the register. The 'text' may contain the RETURN key, which is expanded to carriage return - line feed. If insufficient memory space exists, an error will be given and only part of the 'text' will be inserted. This command is useful for setting up a text register within a command macro.

Notes: If a register needs to be setup from the keyboard, it is often easier to enter the text in Visual Mode and then move it to the text register.

See Also: Function: [MOVE TO TEXT REGISTER]

Example: @RI3/B#EO/ The command "B#EO" is placed into text register 3.

RLr file<ESC> Register Load

Example: RL4 macro1.exc

Description: This command loads text from the file 'file'. The entire file is loaded and the file itself is not affected. If there is insufficient memory space to load the entire file, as much as possible will be loaded and a *BREAK* message will be given. Used to load text files which are then inserted in the text buffer, and to load command macros.

Notes: The text register number should always be specified, but if left out, register 0 will be used.

See Also: Commands: RS, EG
Text registers in Visual and Command Modes.

Example: RL4 macro.exc Loads the file "MACRO.EXC" into text register 4.

RPr Register Print

Example: RP3

Description: This command prints the contents of text register 'r' on the line printer. Allows a hardcopy of the text or command macro in the text register to be made. Also allows a small disk file to be printed after first loading it into a text register. Control characters are expanded and <ESC> is represented as a "$".

Notes: Type CTRL-C to stop.

See Also: Commands: RD, RT

Example: RP5 The contents of text register 5 are printed on the line printer.

RSr file<ESC> Register Save

Example: RS4 macro1.exc

Description: This command saves the contents of text register 'r' in the created file 'file'. The register contents are not affected. If there is insufficient disk space for the entire file, as much as possible is saved and the error "NO DISK SPACE" is given. The error "NO DIR SPACE" results if there is insufficient directory space on the disk. Commonly used to save a section of text in its own disk file, or to save a command macro for later use.

Notes: See note for "RL" command. If an existing 'file' already exists, you will be prompted for confirmation to overwrite it. If there is insufficient disk space to save the register, try deleting some files or insert another disk and give the command again.

See Also: Commands: RL

Example: RS4 macro.exc Saves the contents of text register 4 in the file "MACRO.EXC".

RTr Register Type

Example: RT3

Description: This command types out the contents of text register 'r' on the console. This is commonly used to remind oneself what is in a particular register. <CTRL-S> can be used to pause the display and <CTRL-C> to quit the command. Control characters are expanded and <ESC> is represented as a "$". Any imbedded <CTRL-S> will also pause the display. The "RD" command will dump out a text register without expanding control characters.

Notes: It is frequently easier to insert the text register at the end of the text buffer and view it in Visual Mode.

See Also: Commands: RD

Example: RT5 The contents of text register 5 are displayed on the console.

RU Register Used

Example: RU 50P4$RU

Description: This command displays the number of characters (size) of each text register. Commonly used to see which register hold any text and how much they hold.

Notes: The sum of the displayed values plus the size of the "hidden" help text buffer is the third number displayed by the "U" command. If any of the registers hold text, the status line message "TEXT" is displayed in Visual Mode.

See Also: Commands: U

Example:

RU	Display the sizes of the text registers.
40P3$RU	Save 40 lines of text in register 3 and then check how many bytes are now in the text registers.

R* Register Comment

Example: R* This is a comment in a command macro

Description: This command allows a comment to be placed within command macros. All text through the next <CR><LF> is ignored.

Notes: The "R*" and following comment can appear anywhere in a command macro except in the middle of a text string or file name.

See Also:

Example:

mXAr Value Register Add

Example: XA2 12XA3

Description: This command adds the positive or negative value 'm' to value register 'r'. (If 'm' begins with a '-' the following number is subtracted from the value register) The arithmetic is performed as 16 bit unsigned numbers.

Notes: The 10 value registers are not pre-initialized when VEDIT is invoked.

See Also: Commands: XS, XT

Example:

XA3	This increments (adds "1") value register 3.
60XA4	This adds 60 to value register 4.
-2XA4	"2" is subtracted from value register 4.

nXSr Value Register Set

Example: 0XS2 1200XS3

Description: This command sets value register 'r' to 'n'. 'n' is treated as a 16 bit unsigned integer.

Notes: The 10 value registers are not pre-initialized when VEDIT is invoked.

See Also: Commands: XA, XT

Example:

0XS9	This clears (sets to 00) value register 9.
12XS0	Value register 0 is set to 12.

XTr Value Register Type

Example: XT4 :XT4

Description: This command types value register 'r' in decimal followed by a <CR><LF>. The command may be preceded by a ":", in which case no <CR><LF> will be typed. Currently, this is the only way the value register numbers can be accessed.

Notes: The 10 value registers are not pre-initialized when VEDIT is invoked.

See Also: Commands: XA, XS

Example:

```
10XS2
12XA2
XT2          Value register 2 is typed out, who's
             contents is 22.

YL XT9       The number in value register 9 is
             printed out.
```

YI Route to Text Buffer

Example: YI

Description: This command inserts output to the display screen into the text buffer at the edit pointer. This could be used in conjunction with the "XT" command, for example, to insert numeric information into the text. "0YI" resumes normal display screen output.

See Also: Commands: YL

Example:

```
YI XT3       Insert whatever number is in numeric
             register 3 into the text buffer.
```

YL Route to List Device

Example: YL

Description: This command routes what is sent to the display screen to a printer, or "list device". The command OYL resumes normal display screen output. This command can be used in conjunction with a number of commands, RT, T, EL ED, EV, XT etc., to send information to the printer.

Notes: Do not use the print command "EO" in conjunction with the "YL" command. After an "EO" normal display screen output will resume. Therefore "EO" turns off any "YL" re-routing.

See Also: Commands: YI, T, RT, XT

Examples:

YL ED	Print the directory on the printer.
YL RT6	Print the contents of register 6 on the printer.

YR/YW Display File Name

Example: YR YW

Description: These commands type the names of the input file and output filename. YR types the input file, YW types the output filename. The input file is never listed on the status line in visual mode.

Example:

YR	Lists the input file on the screen.
YW	Lists the output file on the screen.

A P P E N D I C E S

APPENDIX A
CUSTOMIZING VEDIT

CUSTOMIZING VEDIT

WHAT IS CUSTOMIZATION?

Customization is the process of installing VEDIT on your computer in order to adapt it to your particular CRT terminal or video board and your preference in keyboard layout. It also allows you to set various VEDIT parameters according to your applications. The customization is menu driven. You can easily perform some aspects of the customization and leave all other aspects at their previously set or default values. You therefore don't need to understand the entire customization process in order to install VEDIT.

Setting up a new keyboard layout is one aspect of the customization. It allows almost any control character, escape sequence or special function key to be used for the visual mode cursor movements and editing functions. The changeable parameters include the Tab positions, the right margin at which word wrap takes place, and many others. Another aspect is related to your screen size, including the number of lines and columns.

The first part of this appendix gives the step by step instructions for the customization. The later part "Customization Notes" covers some of the customization issues in greater depth.

WHEN IS CUSTOMIZATION NECESSARY?

VEDIT has to be customized before the first time it is used, and can then be customized again, when you have a a new CRT terminal, wish to change some default parameters or just wish to try a new keyboard layout. It is not customized every time you use it. The greatest benefit you receive from the customization process is probably the ability to determine your own keyboard layout, which can utilize any special function keys and accommodate personal preferences.

For some computers, such as the IBM Personal Computer and the IBM Displaywriter, we supply an "up and running" version. The keyboard layout for these preconfigured versions will be listed among the appendices of this manual. Even with a preconfigured version, you will probably want to customize your own later. (We recommend that you use our preconfigured version to gain some experience with VEDIT before customizing your own). If you don't have a preconfigured version of VEDIT, you can create one easily. See "Quick Customization" described under the section "How to Perform Customization". With a CRT version of VEDIT, you generally only need to select the CRT terminal from the menu in order to complete the customization. The keyboard layout will be the "Default Keyboard Layout" which does not assume any function keys. If you have a terminal such as a Televideo 920C or a Zenith Z19, you will probably want to reconfigure the keyboard layout according to the example

layouts for these terminals in Appendix F.

VEDIT is supplied as a disk file with an extension of ".SET", i.e., VEDITZM.SET and VEDITZC.SET, which contain the "prototype" editor to be customized. The customization process does not alter the .SET file, but rather creates a new file with the file extension of ".COM" (or ".CMD" for CP/M-86), which is the runable VEDIT. Depending on your version, you may have several .SET files. Refer to the appendices for the pertinent "Description of Files".

The customization is done with the supplied programs VEDSET.COM for the memory mapped versions, and VDSETCRT.COM for the CRT terminal versions. Since the customization program is fairly easy to run, you will probably run it several times in the first week until you have everything "just right". You can of course also create several configurations of VEDIT, each for a special application. To help remind you of which configuration you are using, you can create a custom signon message for each, which will be displayed when VEDIT is invoked.

CRT and Memory Mapped

There are two primary versions of VEDIT - CRT Terminal versions and Memory Mapped versions. The CRT version supports practically every terminal on the market. The particular terminal being used is selected from a large menu of terminals during customization.

The memory mapped version supports most S-100 and Multibus display boards. Special memory mapped versions are also available for the TRS-80 Models I, II and 16. The memory mapped customization is a little more complex with questions pertaining to the screen's address, cursor types and a few more. The general purpose memory mapped version contains a file on disk which describes the patches necessary to implement bank select.

In addition to the primary versions, there are several specialized versions. One is for the IBM Personal Computer. This version is memory mapped, but skips some parameters, such as the screen address which are fixed by the hardware. It also asks additional questions concerning specific features of the IBM PC, such as display attributes. The appendices describe this in more detail. There is also a version for the PIICEON V-100 and the TDL VDB, both of which are I/O mapped display boards.

HOW TO PERFORM CUSTOMIZATION

STEP 1 - ENTER COMMAND SEQUENCE FROM OPERATING SYSTEM

In order to customize the CRT version of VEDIT you will need the files "VDSETCRT.COM" and "CRT.TBL" on your work disk in addition to the appropriate ".SET" file. For the memory mapped version, you will need the file "VEDSET.COM" and the appropriate ".SET" file. If your disk has more than one ".SET" file, refer to the appropriate "Description of Files" in the appendices for your version of VEDIT.

To begin customization, first type VEDSET or VDSETCRT, followed by the appropriate .SET file name, followed by the file name you wish the editor to be called. (VEDIT is a good choice.) There is no need to give the file extensions to these names. Remember to end the command line with the RETURN key.

Assuming you wish to customize the Z80 CRT version, with a file name of VEDITZC.SET, the customized editor is to be called VEDIT and the files VEDITZC.SET and VDSETCRT.COM are on the currently logged in disk, the command to run VDSETCRT is:

```
VDSETCRT VEDITZC VEDIT
```

A similar command for the 8080 Memory Mapped version would be:

```
VEDSET VEDIT8M VEDIT
```

The command for the TRS-80 Model II Pickles & Trout version is:

```
VEDSET VEDIT2P VEDIT
```

A running VEDIT (a VEDIT.COM or VEDIT.CMD file) may be customized as well. This allows some aspects of the customization to be changed without having to repeat the entire process. A typical command to do this is as follows:

```
VEDSET OLDVEDIT.COM NEWVEDIT
```

where "VEDSET" may also be VDSETCRT, "OLDVEDIT" is the VEDIT you want to change (you need to type the ".COM" or ".CMD" here), and "NEWVEDIT" is the name of the new VEDIT.

If you receive a "Checksum Error", please see the second part of this section for an explanation.

STEP 1.5 - (CRT ONLY). CHOOSE YOUR CRT TERMINAL

VDSETCRT will display a menu of terminals from which you select the number corresponding to your terminal. The list is two screens long; type any key after looking over the first screen. Following the prompt enter the number corresponding to the terminal you are using.

In the rare case that your terminal does not appear on the menu, you have two choices. If you are technically inclined you can change the file "CRT.ASM" which contains the tables corresponding to each of the terminals. This procedure is documented in the file "NEWCRT.DOC". Alternately, contact us for support.

Technically inclined users may wish to read the file "READCRT.DOC" for related information. Hazeltine and Intertube users should also read this file.

STEP 2 - LOOK OVER MAIN MENU TASKS

TASKS:

1) Perform all new keyboard layout
2) Add alternate keys to existing layout
3) Set special characters
4) Set ES and EP parameters
5) Set screen parameters
6) Set other parameters
7) Set signon message
8) Display or print keyboard layout
9) Customization complete; return to operating system

Tasks (1) and (2) are used to determine the keyboard layout, task (8) can print the current keyboard layout, task (7) sets the signon message and (9) writes the customized VEDIT out to disk. The remaining tasks change the various parameters. The prompts for many of these are followed by a number in parentheses, which is a suggested value. To use the suggested value you must type it in, there in NO DEFAULT value. Questions with a numeric answer also require a RETURN after the answer. To ignore input for a particular question, type either the RUBOUT (DELETE) key or a CTRL-U. After each task is performed, the program returns to the main menu. At this point another part of the customization can be performed or a previous step repeated if a mistake was made. Typing a CTRL-C from the main menu aborts the customization.

Quick Customization

If you have the CRT version of VEDIT, the normal screen size of 24 by 80 and wish to bring VEDIT up quickly with the "Default Keyboard Layout", you need only to select your terminal from the CRT terminal menu and then immediately select task (9) in the menu to complete the initial customization. (The default memory size parameters will work well in any size system.) If you like, you can go ahead and do that now, and read the rest of this section later.

If you have a memory mapped system or a CRT terminal with a size other than 24 by 80, you will need to also perform task (5) in the customization. When you are ready to try out something other than the "Default Keyboard Layout" you can perform task (1).

If task (1) is not performed, the resulting editor will respond to the control codes in the "Default Keyboard Layout". Similarly, if tasks (3) through (6) are not performed, the editor will be setup with the parameters in the "Default CRT Customization".

Note: If you have a Televideo, a Z19, an IBM 3101 or other CRT terminal for which we supply a special keyboard layout, you must perform task (1) to give you the keyboard layout. Just selecting the terminal will NOT give you the special layout.

STEP 3 - TASK 1: PERFORM ALL NEW KEYBOARD LAYOUT

ENTER ESCAPE MODE CHARACTER #1

If you choose to use escape sequences, or your keyboard produces escape sequences with special function keys, type the escape character, or the function key lead-in character, most commonly ESC. Else type RETURN, which will then also slip the remaining questions about escape characters.

ENTER ESCAPE MODE CHARACTER #2

A second escape mode character may also be specified, typically for other function keys. If not needed, type RETURN. (Type "CTRL-A" for Televideo terminal).

ENTER COMMON 2ND CHARACTER #1 IN ESCAPE SEQUENCE

Simply answer with a RETURN if you are typing escape sequences in by hand. (A RETURN will skip the next question.) However, some terminal's special function keys send 3 character escape sequences where the second character is always the same and should be ignored. In this case type in the second character. (A "?" for the Zenith Z19) (A capital letter "O" for the DEC

VT-100)

ENTER COMMON 2ND CHARACTER #2 IN ESCAPE SEQUENCE

Some terminals, particularly ANSI standard ones, have a second character which should be ignored in the second character position. Type the character, or if there is no such second character, type RETURN. (A "[" for the DEC VT-100) (RETURN for Z19, Televideo, etc)

UPPER/LOWER CASE ESCAPE SEQUENCES EQUIVALENT (Y/N) ?

If you answer NO, the editor will make a distinction between, for example, ESC H and ESC h. This is annoying if you hand type escape sequences and you should answer with a "Y". However, the function keys on terminals such as a Televideo send escape sequences which distinguish between upper and lower case. Here you would have to answer "N".

TYPE CONTROL CHARACTERS FOR ....

When prompted for each visual operation, you may press a special function key, a control character or enter an escape sequence. The control codes or escape sequences are displayed as you type them in. Use Task (8) to print out the final keyboard layout for your reference. Disallowed characters are the normal displayable characters. Typing one of these will give an error and a reprompt. If you inadvertently attempt to use the same key code for a second operation, an error and a reprompt for the operation will be given. If you do not want to use a particular function, just type RETURN to ignore the function. Specifically, you will probably want to use either [SET INSERT MODE] and [RESET INSERT MODE] or [SWITCH INSERT MODE], but not all three functions. You probably won't use [RESTART], since the function is also available in command mode. Otherwise choose something for [RESTART] which you are very unlikely to hit by mistake. Don't confuse [TAB CURSOR] with the tab character, since it is a cursor movement operation. If you make a mistake, just type RETURN for the rest of the functions and perform this task again.

NOTES: DO NOT ATTEMPT TO USE <CTRL-M> FOR ANY FUNCTIONS, SINCE <CTRL-M> IS THE SAME AS THE RETURN KEY.

The "Left Arrow" on some terminals, especially Televideos, produces the same code as the BACKSPACE key. Therefore, the [BACKSPACE] function needs to be assigned to another key.

It is often convenient to use escape sequences consisting of ESC and a digit for some functions. A strip of cardboard can then be placed above the digits on the keyboard to indicate their functions.

STEP 4 - TASK 2: ADD ALTERNATE KEYS TO EXISTING LAYOUT

Task (2) allows you to use alternate control codes for any of the editing functions. For example, your keyboard may have cursor keys which you have customized as the four basic cursor movements in VEDIT. However, out of habit you are still using CTRL-S, CTRL-D, CTRL-E and CTRL-C to move the cursor. You can select task (2) to enter any such alternate control codes to use for any editing function. Type the RETURN key for those functions you don't wish to invoke by an alternate control sequence.

When running task (2) you should answer the escape character questions the same way as you did for task (1).

Task (2) can also be used to specify the initial control code to use for an editing function if none was specified in task (1), i.e., you ignored the function by typing a RETURN for it. The functional difference between tasks (1) and (2), is that task (1) first clears out any existing keyboard layout, while task (2) builds on the existing layout.

STEPS 5 -> 10 - SET NON-KEYBOARD PARAMETERS:

Answer questions in decimal or hexadecimal as prompted, then hit RETURN. Type a<CTRL-U> or the DELETE (RUBOUT) key to repeat the question.

THERE ARE NO DEFAULT SETTINGS, SO ALWAYS ENTER A VALUE.

STEP 5 - TASK 3: SET SPECIAL CHARACTERS

3.1) HEX CODE FOR SCREEN CONTINUATION CHARACTER (2D)

This is the line continuation indicator used in Visual Mode in reserved column 0. Most common is a hyphen (code 2D hex) or reverse video hyphen (code AD hex).

3.2) HEX CODE FOR COMMAND ESCAPE CHARACTER (1B)

This is the command mode Escape character which should be the "ESC" or "ESCAPE" key (code 1B hex), if your keyboard has it. If your keyboard doesn't have an ESC key, choose another control character, perhaps CTRL-Z, (code 1A hex).

3.3) HEX CODE FOR COMMAND ITERATION LEFT BRACKET (5B)

3.4) HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D)

The Command Iteration Brackets are those which delimit iteration macros --- groups of Command Mode commands. This manual represents these as "[" and "]" with hex codes of 5B and 5D. You may prefer to use "<" and ">" with hex codes of 3C and 3E. Use either set, but it is convenient if your keyboard produces one set without needing the SHIFT key.

3.5) HEX CODE FOR SEARCH PATTERN MATCH CHARACTER (7C)

The initial character of all pattern matching codes is shown in this manual as a "¦", (code 7C hex). If your keyboard does not have a "¦", you will need to change it to some other little used character, perhaps " " (code 7E hex).

3.6) (Memory Mapped Only)
HEX CODE FOR CURSOR CHARACTER (5F)

This is the character used as the blinking "underline" cursor. While normally the underline character (code 5F hex), some users, particularly those with a Sorcerer, may wish to try a hex code of "7F" which is commonly a solid block.

3.7) (Memory Mapped Only)
HEX CODE FOR SCREEN CLEAR CHARACTER (20)
HEX CODE FOR STATUS LINE CHARACTER (2D)
HEX CODE FOR TAB EXPAND CHARACTER (20)

VEDIT normally clears the screen with spaces (code 20 hex), uses a '-' (code 2D hex) on the status line and displays tab characters with spaces. These may be changed for special applications, or if your display requires other characters. For example, the Polymorphic VTI requires that Bit 8 be set for normal characters. Therefore, the character codes would be "A0", "AD" and "A0" respectively.

STEP 6 - TASK 4: SET ES SWITCHES AND EP PARAMETERS

This task selects the default values for these parameters. They can be changed while running VEDIT by using the ES and EP commands. All numeric values are in decimal.

4.1) EXPAND TAB WITH SPACES (0 = NO, 1= YES) (0)

Instead of inserting the tab character into the file, spaces to the next tab position are inserted when the [TAB CHARACTER] function is typed. This is useful if another program interacting with your file doesn't interpret tab characters at the same tab positions. Since many spaces use up extra disk space, don't enable this switch unless you need to.

4.2) AUTO-BUFFERING IN VISUAL MODE
(0=NO, 1=YES, 2=AND BACKWARD) (1)

Auto-buffering is described in section 1. of this manual. You may select no auto-buffering "0", auto-buffering only in the forward direction "1", or auto-buffering in both the forward and backward direction "2". Consider the advantages and disadvantages of backward disk buffering before selecting option "2". We recommend option "1" with floppy disk systems.

4.3) BEGIN IN VISUAL MODE (0=NO, 1=YES) (1)

This determines whether VEDIT starts in Visual or Command Mode. We suggest you set this switch to "Yes".

4.4) POINT PAST TEXT REGISTER INSERT (0=NO, 1=YES) (1)

This determines whether the cursor (or Edit Pointer in Command Mode) will be positioned at the beginning or the end of text inserted from a text register. We suggest that you initially set this switch to "Yes". After some practice with the text registers you will know which way you prefer it.

4.5) IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH
(0=NO, 1=YES) (1)

This determines whether the difference between upper and lower case letters is ignored. We suggest you set this to "Yes". A search for "the" will then also find "The", "THE", etc.

4.6) CLEAR SCREEN ON VISUAL EXIT (0=NO, 1=YES) (0)

This determines whether the screen is cleared when Visual Mode is exited to Command Mode. For most applications you will want to answer "No".

4.7) REVERSE UPPER & LOWER CASE (0=NO, 1=YES) (0)

This determines whether all letters typed on the keyboard will be reversed with regard to upper and lower case, i.e., upper case letters are converted to lower case and vice versa. Only in very unusual situations would you want to set this switch on, so set it off. For the TRS-80 Model I, you should set this switch on, since the keyboard reverses upper and lower case.

4.8) IGNORE SEARCH ERRORS (0=NO, 1= YES) (0)

This switch should normally be off. Otherwise there will be no message if a Find or Substitute is unsuccessful. This switch can be set with the ES command prior to executing some types of command macros.

4.9) USE EXPLICIT TEXT DELIMITERS (0=NO, 1=YES) (0)

This switch, if set ON, allows you to delimit each string in commands such as Substitute or Find with any character. The most commonly used ones are "/", ";", or ";", but any character may be used.

We suggest turning this switch OFF initially because almost none of our examples use this feature. It may be set with the ES command before you begin issuing other commands.

4.10) (Memory Mapped Only)
CURSOR TYPE (0, 1, 2) (1)

This parameter determines the cursor type in memory mapped versions. The cursor types are 0=Blinking Underline, 1=Blinking Reverse Video Block, 2=Solid Reverse Video Block. Most users seem to prefer type "1", but you must use "0" if your display does not produce reverse video. Special applications including the IBM PC and the SSM VB-3 allow a cursor type of 3=Attribute.

4.11) (Memory Mapped Only)
CURSOR BLINK RATE (10 - 100) (See Prompt)

This determines the memory mapped cursor's blink rate. Start with the value suggested by the VEDSET prompt. A smaller number causes the cursor to blink faster.

4.12) INDENT INCREMENT (1 --20, SUGGEST 4)

This determines the "Indent Increment". A value of 4 is common when structured programming languages are being used.

4.13) LOWER CASE CONVERT
(0=NO, 1=YES, 2=CONDITIONAL, 3=REVERSE) (0)

This parameter is useful for assembly language programs. If you choose "0", no conversion will occur. If you choose "1", all lower case keyboard character will be converted to upper case. If "2" is chosen, all characters entered to the left of the "Conditional Conversion Character" will be converted to upper case. For example, assembly language uses ";" as a comment delimiter. To the left of the ";" lower case letters are converted to upper case. To the right of the "; " in the comment field, no conversion is done. Option "3" is similar to "2", except that the case of letters is reversed to the left of the "Conditional Conversion Character".

4.14) DECIMAL CODE FOR CONDITIONAL CONVERSION CHARACTER (59)

This is the "Conditional Conversion" character used when the previous parameter is set to "2" or "3". A value of "59" decimal, makes ";" the Conditional Character.

4.15) LINE AND COLUMN DISPLAY
(0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) (3)

This determines whether the Visual Mode status line will display the line number and column position the cursor is on. It is usually useful to know both.

4.16) RIGHT MARGIN FOR WORD WRAP IN DECIMAL (0=OFF)

The Word Wrap column defines the right margin column. A value of 00 turns Word Wrap off. Words typed beyond the right margin will be wrapped to the next line. The right margin is also used for the [FORMAT PARAGRAPH] function. The right margin can be greater than the screen line length. If VEDIT is being used primarily for word processing, you may wish to enter a value like "70".

4.17) HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1)

This is a technical topic and is described under "High Bit Characters Support" A value of "1" allows high bit (Bit 8) characters on input, but displays them in reverse video

(when possible) after stripping their high bit. This is suggested for most CRT terminals. A value of "3" allows high bit characters on input and causes VEDIT to display them unmodified. This allows accessing alternate character sets and graphics on the IBM PC, NEC APC and some other machines. The value of "7" additionally allows high bit characters, which are not used in the keyboard layout, to be inserted into the text.

4.18) CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)

This determines how the cursor will position itself past the end of lines. A value of "0" causes the cursor to only position itself at "real" characters, i.e. never past the end of a line. A value of "1" allows the cursor past the end of a line during vertical cursor movement. A value of "2" causes VEDIT to pad with spaces when the user attempts editing with the cursor past the end of a line.

4.19) VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254) (200)

This determines the maximum right margin with horizontal scrolling. Lines longer than this value on wrapped on the screen as "continuation lines". Setting this value to or less than the screen line length disables horizontal scrolling.

4.20) HORIZONTAL SCROLL INCREMENT (1 - 100) (20)

This determines how much the screen scrolls sideways when [SCROLL RIGHT] and [SCROLL LEFT] are typed or when VEDIT scrolls the screen automatically.

STEP 7 - TASK 5: SET SCREEN PARAMETERS

5.1) ENTER NUMBER OF SCREEN LINES IN DECIMAL (24 or 25)

Enter the number of lines on your CRT display. While most terminals have 24 lines, some have a 25th "Status Line". On some of these, it is possible for VEDIT to place its status line on the 25th line. These terminals are marked with a "*" following the terminal's name in the menu. To use the 25th line, answer this question with a "25". Note that the Intertec Intertube II must be specified as having 25 lines. The IBM Personal Computer also has 25 lines. Answer with "24" for the Televideo 950C, since VEDIT cannot

use its status line.

5.2) ENTER LINE MOVEMENT FOR PAGING IN DECIMAL (20)

Enter the number of screen lines you wish [PAGE UP] and [PAGE DOWN] to move through the text by. About 4/5 of the total number of screen lines is suggested, i.e., "12" for a 16 line display and "20" for a 24 line display.

5.3) ENTER TOP LINE FOR CURSOR IN DECIMAL (3)

This sets the top screen line the cursor can normally be on, before the screen will begin to scroll down. This is therefore, the minimum number of lines you will always see before the line you are editing.

5.4) ENTER BOTTOM LINE FOR CURSOR IN DECIMAL (20)

This is similar to the previous step, except that it sets the bottom line range for the cursor. This number must be greater than or equal to the "Top Line for Cursor" setting, and at most be one less than the "Number of Screen Lines", since the very bottom line is only used for status. "4" less than the number of screen lines is a good starting point.

5.5) (Memory Mapped Only)
ENTER SCREEN LINE LENGTH IN DECIMAL (80 or 64)

Enter the number of bytes per screen line your display has. This number must be in the range 20 - 255. It will be 64 or 80 for most Memory Mapped Displays and therefore the same as the next question. However it will be greater for displays which have invisible attribute bytes at the end of each line. The MATROX display board is an example, and requires a value of 128.

5.6) ENTER LENGTH OF DISPLAYED LINE IN DECIMAL (80 or 64)

This is the number of characters per line VEDIT will display. Normally you want this value equal to the screen line length, usually 80 or 64. The TRS-80 Model II and 16, the IBM Personal Computer and the MATROX display board require a value of 80.

5.7) (Memory Mapped Only)
ENTER ADDRESS OF SCREEN IN HEXADECIMAL

For 8080/Z80 versions enter the memory address of the beginning of the video in hexadecimal. Many 16 x 64 boards have an address of CC00 hex. The TRS-80 Model II has an address of F800 hex.

For 8086 versions enter the segment address for the base of the screen.

5.8) (8080/Z80 Memory Mapped Only)
ENTER NUMBER OF VIDEO BOARD INITIALIZATION BYTES

Enter "0" if your board requires no initialization. Otherwise enter a number between "1" and "5" for the number of "data byte"-"port address" pairs needed for initialization. Most memory mapped system need no initialization (including TRS-80 Models I, II and 16). (One exception is the Processor Technology VDM, which requires a "00" output to port "C8" hex, and the SOL-20 a "00" output to port "FE" hex).

ENTER [RUBOUT] OR [CTRL-U] TO START PAIR OVER
ENTER DATA BYTE
ENTER PORT ADDRESS

The specified number of "data byte"-"port address" pairs is entered in hexadecimal with each number followed by RETURN. Typing CTRL-U or RUBOUT will reprompt with the "ENTER DATA BYTE" question for that pair.

STEP 8 - TASK 6: SET OTHER PARAMETERS

6.1) SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO-READ IN BYTES

See the table below for a recommended value depending upon your memory size. The number must be in the range 1024 - 32768. Use RUBOUT or <CTRL-U> if you mistype the number.

MEMORY SIZE	SPARE MEMORY FOR	VALUE FOR TRANSFER
20K	2048	3
24K	2048	4
28K	3072	5
32K	4096	6
36K	5120	7
40K	5120	8
44K	6144	9
48K	6144	10
52K	7168	11
56K	7168	12
60K	8192	13
64K	8192	14

--- Minimum system size = 20K.
--- 1 K byte is a unit of 1024 bytes (1024 = 2 ** 10).
--- For CP/M systems, the memory size is the CP/M size, which should be on your CP/M disk label or displayed when you first boot.
--- Do not make the Spare Memory for Auto Read more than two times larger than the value given in the table or it may produce a non-operational editor. This value represents the number of bytes free in the text buffer AFTER a file larger than available memory space is read. For example, in a 64K system the available memory is about 46K. If the table value of 8192 was chosen and a very large file edited, VEDIT would initially read in the first 38K of the file, leaving 8192 bytes free. This extra space is available for insertion of new material. Use the "U" command to verify actual free space. See "Customization Notes" for more details.

6.2) SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES

Choose the value from column 3 of the above table which corresponds to your memory size. This parameter specifies the amount of the file read into the text buffer when auto-buffering is done. The number entered must be in the range 1 - 32.

6.3) DO YOU WISH TO USE DEFAULT TAB POSITIONS? (Y/N)

The default tab positions are set at every 8th position, for example, 9 17 25 41 49 57 65 73 81 89 etc. This is the most common tab setting; if you change the tabs, the change will apply to VEDIT only. Tab positions may be reset inside VEDIT by using the ET command.

If you enter "N", this prompt is given:

ENTER UP TO 30 TAB POSITIONS IN DECIMAL

Enter the desired tab positions, separating the numbers with spaces or commas and following the last number with a RETURN. Don't be concerned if your input line goes off the right side of your terminal or screen. Note that you need no tab at position 1 and that the positions are counted starting from 1, not 0. You must also specify at least one tab position per screen line and the highest allowed position is 254. Entering a number outside of the range 1 - 254 will give an error and a reprompt of the question. If you make a mistake, type RUBOUT or <CTRL-U> to start the question over.

6.4) (CRT Only)
ENTER DECIMAL VALUE (4mhz = 76, 2mhz = 38)

Enter "76" if your processor speed is 4mhz, "38" if the speed is 2mhz. Interpolate for other processor speeds. This value is used for CRT's which require time delays after control sequences are sent to them and is not critical. It is also used in computing the delay for updating the status line. The maximum value is 255.

6.5) BEGIN IN INSERT MODE (0=NO 1=YES) (0)

During full screen editing, you are either in "Normal" or "Insert" mode. This question lets you select which mode the editor begins in. Answering this question is a matter of personal preference. The status line always indicates which mode you are in.

6.6) USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES
(0 = NO, 1 = YES) (0)

If you want VEDIT to search on the default drive for the VHELP.TXT and optional VEDIT.INI files, answer with a "1".

Normally you should place these files on the same drive as the VEDIT.COM or VEDIT.CMD file.

6.7) ENTER (0 = NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES (0, 1 - 16) (1)

In addition to the default drive, VEDIT can search on any other designated drive for the VHELP.TXT and VEDIT.INI files. Drive A is "1", drive B is "2", etc. Users who keep their system disk with VEDIT, VHELP and the optional VEDIT.INI on drive A should, therefore, answer with "1". If this and the previous question are answered with "0", the help system cannot be accessed.

6.8) SHOULD VEDIT.INI FILE BE EXECUTED (0 = NO, 1 = YES)

If you always or occasionally use the auto-startup facility, answer with "1". If you never use it answer with "0". With a value of "0", VEDIT will not even search for the VEDIT.INI file, which speeds up the invoking process by a little.

6.9) REVERSE VIDEO ON STATUS LINE (0 = NO, 1 = YES) (1)

If your CRT or video display board produces reverse video, answer "1". If you have a CRT terminal which does not produce reverse video, answer "0".

STEP 9 - TASK 7: SET SIGNON MESSAGE

This message will appear briefly whenever you invoke VEDIT. It can be used to help you identify how the particular VEDIT was customized. The message may be up to 64 characters long. An example message might be:

Bob's Televideo 920C, Word Wrap = 70.

STEP 10 - TASK 8: DISPLAY OR PRINT KEYBOARD LAYOUT

Selecting this task results in the following question:

DISPLAY ON PRINTER (0) OR CONSOLE (1) ?

Type a "0" (and RETURN) if you wish to have the keyboard layout printed, or "1" to see it displayed on the console screen. The display will be similar to our example keyboard layout sheets, and will also show any alternate keyboard sequences that may be used for each function. This is a handy way to make a record of the keyboard layout. If you forget the keyboard layout, you can run the customization program on the runable VEDIT file and select Task 8 to print out the keyboard layout. For example, if your VEDIT is in the file VEDIT.COM and your customization program is VDSETCRT.COM you can give the command:

```
VDSETCRT VEDIT.COM JUNK
```

Select Task 8 to print the keyboard layout, and then type CTRL-C to abort the customization process.

STEP 11 - TASK 9: CUSTOMIZATION COMPLETE; RETURN TO OPERATING SYSTEM

This writes the customized VEDIT out to disk.

Customization Notes

This section describes some aspects of the customization in more detail. You do not need read this section in order to get VEDIT up and running. However, once you are more familiar with VEDIT, you will probably want to gain a better understanding of the customization in order to create a more "personalized" version of VEDIT.

VEDIT Checksum

To help insure that your distribution diskette is intact, the customization performs a checksum on the VEDIT file being customized. If there is a fault, a warning error message is given. If you encounter this error make sure that you have copied the files from the distribution diskette properly. If all else fails, try running the customization on the distribution diskette. If this still results in the error, please contact us for an exchange diskette. If you have patched the VEDIT file, this error will result. In this case it can be ignored, and the new VEDIT file will contain a new checksum so that the error will not occur again unless the file becomes modified again.

Keyboard Layout

Determining the desired keyboard layout for the cursor movement and function keys is the first task of the customization. Since it could be a difficult task, several example keyboard layouts are enclosed to help the new user. The best layout will depend to some extent upon your keyboard, especially if you have one with extra keys which produce control codes. If extra keys are available, you may want to allocate them to the most used visual operations such as the cursor movements. The more extra keys you have, the easier it becomes to remember the layout.

If and when you decide to try out your own layout, you will want to avoid placing the keys you least want to hit by accident, such as [Erase Line] or [Home], right next to the cursor movement keys. In the event that you have no or few special keys, most visual operations will involve holding the CONTROL key while you type a letter, or using escape sequences. In this case, the layout may be tight and difficult to organize. One strategy is to use mnemonic letters, such as CTRL-D for [DELETE] and CTRL-U for [UNDO], etc. Another is to arrange the keys in some geometric manner, such as having the cursor movement keys on one side of the keyboard and the visual function keys on the other side. You can also simplify the layout by using escape sequences, especially for functions you do not use often, or don't want to hit by accident. Trying out some combinations on paper is probably the easiest way to accomplish the layout task.

Besides responding to the customary control characters, VEDIT also handles multi character escape sequences. These may be user typed, or may result from pressing a special function key. For example, instead of typing the single character CONTROL-Q, the user may type two characters, i.e. ESC and Q, to perform a visual operation. All escape sequences begin with one of two user defined escape characters (sometimes called Lead-in characters). While the ESC is a common key to use as an escape character, any other ASCII character may be used as the escape character, even displayable ones like "@". The special function keys on some keyboards, like the Heath H19, Televideo 920C and IBM 3101 also send multi character escape sequences. Some terminals, like the IBM 3101, also send a Carriage Return at the end of escape sequences. The keyboard customization detects this automatically and the user need not be concerned with it.

When performing the keyboard customization, it asks the question: "Ignore upper/lower case difference in escape sequences?" If you answer NO to this question, the editor will made a distinction between, for example, "ESC-H" and "ESC-h". Therefore, if you entered the escape sequence with a lower case "h" during customization, the editor would not respond to the escape sequence with an upper case "H". This is annoying if you hand type most of the escape sequences, since at times you may have the SHIFT or a CAPS-LOCK depressed. You would therefore want to answer the question with a YES. However, the function keys on some terminals, such as a Televideo, send escape sequences which distinguish between upper and lower case letters. In this case you will want to answer the question with NO. If you find that you have made a mistake with this question, you can skip performing the entire keyboard customization again, by performing task (2) in the customization, answering this and the other three questions pertaining to escape sequences correctly and simply typing a RETURN for all of the function prompts.

When laying out the keyboard, you may therefore use any combination of control characters, special function keys and escape sequences for the visual operations. Some users will prefer to use function keys and control characters for the most used visual operations, and escape sequences for the less used operations. If escape sequences are used, a key like ESC or FORM FEED is suggested for the escape mode character. Any other character may then follow, including numbers, control characters or even another escape character. Many keyboards have a numeric pad and these numbers can be used in escape sequences. For example, use ESC - 8 for [CURSOR UP], ESC - 2 for [CURSOR DOWN], ESC - 4 for [CURSOR LEFT] and so on. In this case you may wish to attach descriptive labels on top of the numeric keys. An Escape and Control character combination is a good choice for operations you don't want to hit by mistake, like [HOME], [ZEND] or [RESTART EDITOR]. You may use an escape sequence consisting of two escape characters in a row. In fact, if ESC is the escape

character, then "ESC - ESC" is the suggested sequence for the function [VISUAL ESCAPE]. In the unusual case that a displayable character like "@" is used as the escape character, a "@ - @" cannot be used for a visual operation, since in this case, "@ - @" will be treated by VEDIT as the normal "@" character.

While all of this is complicated enough already, there are a few pitfalls to avoid too. (You are well advised to use one of the example keyboard layouts at first.) The only key which is predefined is the RETURN or CR key which is also CTRL-M and cannot be used for any visual operation. The special function keys on some keyboards send a code identical to a control character. You may therefore unintentionally attempt to use the same control code for two visual operations. In this case, VEDSET or VDSETCRT will give an error message and request a new key for that function. Some keyboards have special function keys which send a character with data bit 8 set (sometimes called the "High" or "Parity" bit). These work properly since the VEDIT programs decode all 8 bits.

A Word About Keyboards

With the simplest keyboards, each visual operation will have to be activated by holding the CONTROL key and typing some letter or using an escape sequence. Moving up, keyboards will have keys for Backspace, Tab and Line Feed, which can be used to perform the described function. Some keyboards with a numeric pad can send control codes by holding the SHIFT or CONTROL key and typing one of the pad keys. Numeric pad keys can always be used as part of escape sequences. The pad can then be used for most of the visual operations. In some cases, the keyboard will have many special keys, which send a control code just by typing one of them. In the ideal case, these control codes will be sent with the highest data bit set. (This is Bit 8 and is often called the parity bit. The ASCII standard code does not use Bit 8 and even a "Full ASCII" keyboard will send nothing on Bit 8 or else parity information). Some very special keyboards, usually ones with 70-100 keys on them, use Bit 8 to decode all those keys. Since VEDIT and VEDSET decode all 8 data lines from the keyboard, these fancy keyboards can be used to their full advantage.

Screen Size Parameters

VEDIT can be customized for any screen size up to 70 lines by 250 columns. To set these parameters you need to know the number of lines and the number of characters per line that your CRT terminal or video display board produces. 16 x 64 and 24 x 80 are the most common values. You also have the choice of how many columns on a line are actually used. You want to use all of them, unless you have a special

application or unusual hardware.

For the memory mapped versions, you also need to know the beginning address of the display board in memory in hexadecimal and whether it requires any data bytes output to a port to initialize it. For example, many 16 x 64 boards have an address of CC00 hex. Most of these 16 x 64 boards do not need any initialization, one exception being the Processor Technology VDM board, which should have a 00 output to Port C8 hex. (The SOL-20 requires a 00 output to Port FE hex).

Memory Size Dependent Parameters

The first parameter "Spare Memory for Auto-Read" determines how many bytes of memory are free after VEDIT does an auto-read (such as following an EB command). This size must be specified between 1024 and 32768. A reasonable size is about 1/4 of the size of the text buffer for small systems and a little less for large systems. Choosing a 1K (1024 byte) multiple makes the disk read/write work a little bit faster.

In particular, do not make this value more than 2 times larger than the value in the table, or you may produce a non-operational editor. This value is NOT the amount of memory VEDIT will use for the text buffers, since VEDIT always sizes memory and uses all that is available. Rather, this value is the number of bytes that is free in the text buffer after a file is read which is larger than the available memory space. For example, in a 64K system the available memory is about 46K. If the table value of "8192" was used, and a very large file edited, VEDIT would initially read in only the first 38K of the file, leaving "8192" bytes free. This can be verified with the "U" command.

The second parameter "Size of File Transfers" specifies the size of file transfers during auto-buffering and for the 'N' command. For normal use, a value about 1/3 the size of the text buffer is good. (Specifying a value larger than one half the maximum text buffer size may create a non-working version of VEDIT.) When auto-buffering is initiated, an attempt is made to append this number of K bytes to the end of the text. If there is insufficient memory space for appending this many bytes, this many bytes are written from the beginning of the text buffer to the output file. An auto-read is then performed which reads in the rest of the input file, or until the memory is filled to within the number of spare bytes specified by "Spare Memory for Auto-Read".

APPENDIX B
QUICK COMMAND REFERENCE

'n' denotes a positive number. (# represents 32767)
'm' denotes a number which may be negative to denote backwards in the file.
'r' denotes a digit "0 - 9" specifying a text or numeric register 'text','string', and 's1' denote text strings
'file' is a file name in the normal CP/M (MSDOS) format with optional drive and extension. Must end in RETURN or <ESC>.

nA	Append 'n' lines from the input file. (OA)
-nA	Read 'n' lines back from output file. (-OA)
B	Move the edit pointer to text beginning.
mC	Move the edit pointer 'm' characters.
mD	Delete 'm' characters from the text.
nFstring<ESC>	Search for 'n'th occurrence of 'string' in buffer
Gr	Insert the contents of text register 'r'.
H	Display Help file VHELP.TXT.
Itext<ESC>	Insert the 'text' into the text buffer.
mK	Kill 'm' lines.
mL	Move the edit pointer by 'm' lines.
Mr	Execute text register 'r' as a command macro.
nNstring<ESC>	Search for 'n'th occurrence of 'string' in file.
mPr	Put 'm' lines of text into text register 'r'.
nSs1<ESC>text<ESC>	Search for and change 's1' to 'text', 'n' times
mT	Type 'm' lines on console.
U	Print # of unused, used and text register bytes.
V	Go into visual mode.
nW	Write 'n' lines to the output disk file. (OW)
-nW	Write last 'n' lines to "R" file.(-OW)
Z	Move edit pointer to end of text buffer

SPECIAL CHARACTERS

<CTRL-Q>	Literal Character. Next char. is taken literally.
@	Precedes F, I, N, S commands to indicate explicit delimiter instead of <ESC>.
:	Precedes F, N, S commands to suppress search error message. Precedes XT, YR and YW commands to suppress CRLF.
_	Precedes F, S commands to perform search to end of file. Note: "_F" is equivalent to "N". Precedes B, L, Z commands to global operation on entire file.
#	Represents maximum positive number 32767. Signifies "forever" or "all occurrences of".

EXTENDED COMMANDS

EA	Save file and reload. (EX and EB).	
EBfile	Open 'file' for Read & Write, perform an auto-read.	
EC	Change disks for reading or write error recovery.	
ED	Display disk directory. Opt. drive spec. and "?".	
EF	Close the current output file.	
EGfile[line range]	Insert the specified line number range of the file 'file' into the text buffer at the edit position. If no line range, insert entire file.	
nEI	Insert the character whose decimal value is 'n'.	
EKfile	Erase (kill) the file 'file' from the disk.	
ELfile[line range]	Display with line numbers the specified line number range of the file 'file'.	
mEO	Print 'm' lines on the line printer. (0E0)	
EP n m	Change the value of parameter 'n' to 'm'.	
1	Cursor type	(0, 1 or 2)
2	Cursor blink rate	(10 - 100)
3	Indent Increment	(1 - 20)
4	Lower case convert	(0, 1 or 2)
5	Conditional convert character	(32 - 126)
6	Display line and column number	(0, 1, 2 or 3)
7	Word Wrap column (0 = Off)	(0 - 255)
8	Bit 7 Suppressed (0/input 1/Output)	(0 - 7)
9	Cursor positioning option	(0 - 2)
10	Virtual line length with scrolling	(40 - 255)
11	Horizontal scroll increment	(1 - 100)
EQ	Quit the current edit session, without saving file	
ERfile	Open the file 'file' for input.	
ES n m	Change the value of switch 'n' to 'm'.	
1	Expand Tab with spaces	(0=NO 1=YES)
2	Auto buffering in visual mode	(0=NO 1=YES 2=BACK)
3	Start in visual mode	(0=NO 1=YES)
4	Point past text reg. insert	(0=NO 1=YES)
5	Ignore UC/LC search distinction	(0=NO 1=YES)
6	Clear screen on visual exit	(0=NO 1=YES)
7	Reverse Upper and Lower case	(0=NO 1=YES)
8	Suppress search errors	(0=NO 1=YES)
9	Explicit string terminators	(0=NO 1=YES)
ET	Set new tab positions.	
EV	Print the VEDIT version number.	
EWfile	Open the file 'file' for output. Create Backup.	
EX	Normal exit back to Operating System after writing output file.	
EY	Finish writing and close output file.	
EZ	Quit the current edit session, remain in VEDIT.	

TEXT REGISTER COMMANDS

R*	Treat following line as a comment
RDr	Dump contents of register 'r' on console.
RIrtext<ESC>	Insert text into register 'r'.
RLrfile	Load register 'r' from file 'file'.
RPr	Send contents of register 'r' to line printer
RSrfile	Save contents of register 'r' in file 'file'.
RTr	Type contents of register 'r' on console.
RU	Display size of each text register.

NUMERIC REGISTER COMMANDS

nXAr	Add 'n' to value in register 'r'.
nXSr	Set value of register 'r' to 'n'.
XTr	Type (decimal) value of 'r'.

MISCELLANEOUS COMMANDS

YI	Route following console output to text buffer.
YL	Route following console output to printer.
YR	Type the "read" input file name on console.
mYS	Strip 'm' lines of Bit 7.
YTtext<ESC>	Type 'text' on console on console.
YW	Type the "write" output file name.

SEARCH PATTERN MATCHING CODES

¦A	Match any Alphabetic letter, upper or lower case
¦B	Match a Blank - single space or tab
¦C	Match any Control character
¦D	Match any numeric digit - "0" - "9"
¦L	Match Line terminator - Line Feed, Form Feed or EOF
¦M	Multi - match any sequence of zero or more characters
¦N	Match any except following character or "¦ code"
¦R	Match any Alphanumeric - a letter or a digit
¦S	Match any Separator - not a letter or digit
¦U	Match any Upper case letter
¦V	Match any Lower case letter
¦W	Match White space - single or multiple spaces or tabs
¦X	Match any character. (Equivalent to old "¦" wildcard)
¦¦	Use "¦¦" when you need to search for a "¦"

APPENDIX C
ERROR MESSAGES

VEDIT types a message on the console when you should be notified of an unusual or special condition. All messages are descriptive, and you should not normally have to refer to this appendix in order to understand the message or error. The messages fall into three categories: fatal errors, non-fatal errors and other messages. Fatal errors result in an abort of the disk operation being performed and a return to Command Mode. These are caused by certain disk errors described below. The non-fatal errors usually just signify that a typo was made or that some small detail was overlooked. These only result in a message and you can try again.

FATAL ERRORS

CLOSE ERROR — The output file could not be closed. This is a very unusual condition, but may occur if the disk becomes write protected.

NO DIR SPACE — There was no directory space left for the output file. Refer to the section on disk write error recovery.

NO DISK SPACE — The disk became full before the entire output file was written. As much of the output file as possible was written. Refer to the section on disk write error recovery.

READ ERROR — An error occurred reading a file. This error should never occur, since The operating system itself normally gives an error if there was a problem reading the disk.

REV FILE OPEN — You cannot change disks because the ".R" file is open while performing backward disk buffering.

NON-FATAL ERRORS

BAD FILE NAME — The file name you gave does not follow the MSDOS or CP/M conventions.

BAD PARAMETER — Something was specified wrong with your "EI", "EP", "ES" or "ET" command.

CANNOT FIND... — The specified search string could not be found. This is the normal return for iteration macros which search for all occurrences of a string.

CANNOT OPEN TWO — You cannot have two output files open and there is already one open. Also given if an output file is open at the time of an "EC" command. Perhaps you want to close it with the "EY" command.

DISK IN USE — You cannot perform an "EC" command because the disk is being used by another program or user. See section on Multi-Tasking Operating Systems.

DISK IS R/O — The file you are trying to edit or write to already exists and and has a "Read Only" attribute. Your command is canceled. You will have to return to the operating system and change its attribute before you can edit it.

FILE NOT FOUND — The file you wanted to open for input does not exist. Maybe you specified the wrong drive.

FILE NOT OPENED — This message follows another message and reminds you that your attempted disk operation was canceled. It follows the "DISK IS R/O" message. Also follows an operating system error message if you attempted to open a file which is in use by another program or user. See section on Multi-Tasking Operating Systems.

INVALID COMMAND — The specified letter is not a command.

MACRO ERROR — Your macro attempted to change the contents of a text register which is currently executing as a command macro.

NESTING ERROR — You cannot nest macros deeper than 8 levels.

NO INPUT FILE — There is no input file open for doing a read or append operation.

NO OUTPUT FILE — There is no output file open for doing a write, a close or an exit with the "EX" command. If you have already written out the text buffer and closed the output file, exit with the "EQ" command. You must have an output file open when performing global file operations.

OTHER MESSAGES

BREAK

The command execution was stopped because insufficient memory space remained to complete the command (I, S, G, P and EG). For the "I", "S" and "EG" commands, as much text as possible was inserted. For the "G" and "P" commands, no text at all was copied or inserted. The message is also printed when command execution is stopped because you typed <CTRL-C> on the keyboard in Command Mode or while printing.

INSERT NEW DISK AND TYPE [RETURN]

This is the normal prompt for inserting a new disk with the "EC" command.

NEW FILE

The file specified with the EB command or with the invocation of VEDIT did not exist on disk and a new file has been created. If you typed the wrong file name, you may want to start over by issuing the "EZ" command.

QUIT (Y/N)?

This is the normal prompt following the "EQ" or "EZ" command. Type "Y" or "y" if you really want to quit, without saving the file, otherwise type anything else. To avoid this prompt when quitting, enter "EQY" or "EZY" instead of "EQ" or "EZ".

PLEASE WAIT
FOR DISK

Temporary message on status line to indicate that a disk operation is being performed during Visual Mode. Some systems will loose key-strokes if you type during disk operations.

PRINTING <CTRL-C>
TO ABORT

Normal message anytime text is being printed. Reminds you that you can press <CTRL-C> to stop the printing.

WAITING FOR PRINTER
<CTRL-C> TO ABORT

VEDIT is waiting for another program or user to release the printer before it can begin printing. Message changes to normal printing message once printing begins. See section on Multi-Tasking Operating Systems.

APPENDIX D
EXAMPLE KEYBOARD LAYOUTS
GENERAL VERISIONS

DESCRIPTION OF FILES ON DISK

General 8080 / Z80 / 8086 Versions

The files actually supplied on your diskette depend upon which version and package you purchased. You will have to perform the customization process, described in the manual, to produce a runnable version of VEDIT.

VDSETCRT.COM	The program used to perform the customization for the CRT versions. The manual describes the use of this program and the "SET" files below.
VEDSET.COM	The program used to perform the customization for the memory mapped versions.
VEDITZC.SET	File for producing the Z80 CRT version.
VEDIT8C.SET	File for producing the 8080 CRT version.
VEDITZM.SET	File for producing the Z80 Memory mapped version.
VEDIT8M.SET	File for producing the 8080 Memory mapped version.
VEDIT86C.SET	File for producing the 8086 CRT version

PERSONAL KEYBOARD LAYOUT

"ESCAPE MODE CHARACTER #1"
"ESCAPE MODE CHARACTER #2"
"COMMON 2ND CHARACTER #1..."
"COMMON 2ND CHARACTER #2..."
"UPPER/LOWER CASE ESCAPE..."

[HOME] -
[ZEND] -
[CURSOR UP] -
[CURSOR DOWN] -
[CURSOR RIGHT] -
[CURSOR LEFT] -
[BACK TAB] -
[TAB CURSOR] -
[ZIP] -
[NEXT LINE] -
[LINE TOGGLE] -
[SCROLL UP] -
[SCROLL DOWN] -
[SCROLL RIGHT] -
[SCROLL LEFT] -
[PREVIOUS WORD] -
[NEXT WORD] -
[PREVIOUS PARAGRAPH] -
[NEXT PARAGRAPH] -
[PAGE UP] -
[PAGE DOWN] -
[SCREEN TOGGLE] -
[BACKSPACE] -
[DELETE] -
[ERASE TO END OF LINE] -
[ERASE LINE] -
[DEL PREVIOUS WORD] -
[DEL NEXT WORD] -
[UNDO] -
[TAB CHARACTER] -
[NEXT CHAR LITERAL] -
[SET INSERT MODE] -
[RESET INSERT MODE] -
[SWITCH INSERT MODE] -
[REPEAT] -
[INDENT] -
[UNDENT] -
[FIND] -
[REPLACE] -
[CANCEL] -
[COPY TO TEXT REGISTER] -
[MOVE TO TEXT REGISTER] -
[INSERT TEXT REGISTER] -
[PRINT TEXT] -
[SET TEXT MARKER] -
[GOTO TEXT MARKER] -
[FORMAT PARAGRAPH] -
[VISUAL ESCAPE] -
[VISUAL EXIT] -
[RESTART EDITOR] -

DEFAULT CRT CUSTOMIZATION

The customization session below indicates the default values supplied in the generic versions of VEDIT. The two values depending upon memory size are based on a 40K system, however, they may be used for larger size systems too. For clarity sake, each reply below is preceded by "--", which does not appear is the actual customization.

```
3.) HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- 2D
    HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
    HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
    HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
    HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C

4.) EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
    AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
    BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
    POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
    IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
    CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
    REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
    IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 0
    EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

    INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
    LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
    DECIMAL CODE FOR CONDITIONAL CONVERSION CHAR. (59)   -- 59
    LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
    RIGHT MARGIN FOR WORD WRAP IN DECIMAL (0=OFF)        -- 0
    HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1)  -- 1
    CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
    VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
    HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.) ENTER NUMBER OF SCREEN LINES IN DECIMAL      -- 24
    ENTER LINE MOVEMENT FOR PAGING IN DECIMAL    -- 20
    ENTER TOP LINE FOR CURSOR IN DECIMAL         -- 3
    ENTER BOTTOM LINE FOR CURSOR IN DECIMAL      -- 20
    ENTER LENGTH OF DISPLAYED LINE IN DECIMAL    -- 80

6.) SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 6144
    SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 8

    DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

    ENTER DECIMAL VALUE  (4MHZ = 76, 2MHZ = 38) -- 76
    BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

    USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 1
    ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
    SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

    REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

Note: Most terminals, except Televideos, can use a value of "AD" instead of "2D" in task 3.1. This will cause the continuation line character to be displayed in reverse video.

DEFAULT KEYBOARD LAYOUT

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)
[HOME]	ESC - H	
[ZEND]	ESC - Z	
[CURSOR UP]	[CTRL-E]	
[CURSOR DOWN]	[CTRL-C]	
[CURSOR RIGHT]	[CTRL-D]	
[CURSOR LEFT]	[CTRL-S]	
[BACK TAB]	[CTRL-A]	
[TAB CURSOR]	[CTRL-F]	Useful for fast cursor movement.
[ZIP]	[CTRL-G]	
[NEXT LINE]	[CTRL-J]	Not needed in tight layouts
[LINE TOGGLE]	NOT USED	Or use instead of [ZIP]
[SCROLL UP]	[CTRL-Q]	
[SCROLL DOWN]	[CTRL-Z]	
[SCROLL RIGHT]	[CTRL-Y]	
[SCROLL LEFT]	[CTRL-T]	
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	ESC - W	
[NEXT PARAGRAPH]	ESC - X	
[PAGE UP]	[CTRL-W]	
[PAGE DOWN]	[CTRL-X]	
[SCREEN TOGGLE]	[CTRL-N]	
[BACKSPACE]	[CTRL-H]	Or use BACK SPACE key.
[DELETE]	[DEL]	Or called RUBOUT.
[ERASE TO END OF LINE]	[CTRL-L]	Also called [EREOL] in manual.
[ERASE LINE]	ESC - L	
[DEL PREVIOUS WORD]	ESC - V	
[DEL NEXT WORD]	ESC - B	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[CTRL-I]	Or use TAB key.
[NEXT CHAR LITERAL]	ESC - Q	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CTRL-K]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	ESC - F	
[REPLACE]	ESC - R	
[CANCEL]	ESC - O	
[COPY TO TEXT REGISTER]	ESC - C	
[MOVE TO TEXT REGISTER]	ESC - M	
[INSERT TEXT REGISTER]	ESC - I	
[PRINT TEXT]	ESC - P	
[SET TEXT MARKER]	ESC - S	
[GOTO TEXT MARKER]	ESC - G	
[FORMAT PARAGRAPH]	ESC - B	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	ESC - E	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

VEDIT NOTES FOR HEATH / ZENITH Z19 / Z19 TERMINALS

The example keyboard layout assumes that the terminal is in the "Heath" mode and not in the ANSI mode. While VEDIT automatically puts the keypad into "Alternate Keypad Mode", VDSETCRT does not, and for the customization you must enter the Alternate Keypad Mode by going into Local Mode and typing ESC and '='.

You may experience extra characters appearing when using the cursor UP and DOWN keys, especially in conjunction with the REPEAT key. This is caused by the function keys sending their multi character codes at 9600 Baud, which is too fast for non-interrupt driven software. This is best solved by making your system interrupt driven. If this is not possible, implement the cursor movements with control characters and use the keypad for other functions. It may help to reduce the entered value in Task 6.4 (relating to processor speed) during customization, or to use the "Shifted Keypad Mode".

If you are using your terminal at the undocumented 19,200 baud, you will need to modify the CRT.TBL file (see NEWCRT.DOC") and increase the delays for EOS, EOL, INSERT LINE and DELETE LINE. The cursor addressing will also require a delay of about 4 milliseconds.

EXAMPLE KEYBOARD LAYOUT FOR THE H19

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	?	
"COMMON 2ND CHARACTER #2..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	0	(1 = YES, 0 = NO)

[HOME]	ESC - CTRL-H	
[ZEND]	ESC - CTRL-Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[BLUE]	
[TAB CURSOR]	[RED]	Useful for fast cursor movement.
[ZIP]	[WHITE]	
[NEXT LINE]	[LINE FEED]	
[LINE TOGGLE]	[CTRL-A]	
[SCROLL UP]	SHIFT [Up Arrow]	
[SCROLL DOWN]	SHIFT [Down Arrow]	
[SCROLL RIGHT]	SHIFT [Right Arrow]	
[SCROLL LEFT]	SHIFT [Left Arrow]	
[PREVIOUS WORD]	[CTRL-D]	
[NEXT WORD]	[CTRL-F]	
[PREVIOUS PARAGRAPH]	SHIFT [IC]	
[NEXT PARAGRAPH]	SHIFT [IL]	
[PAGE UP]	[IC]	
[PAGE DOWN]	[IL]	
[SCREEN TOGGLE]	[CTRL-W]	
[BACKSPACE]	[BACK SPACE]	
[DELETE]	[DC]	
[ERASE TO END OF LINE]	[DL]	Also called [EREOL] in manual.
[ERASE LINE]	SHIFT [DL]	
[DEL PREVIOUS WORD]	[CTRL-S]	
[DEL NEXT WORD]	[CTRL-G]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	ESC - Z	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[ENTER]	"ENTER" on Keypad
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	[F1]	
[REPLACE]	[F2]	
[CANCEL]	[F3]	
[COPY TO TEXT REGISTER]	[F5]	
[MOVE TO TEXT REGISTER]	[ERASE]	
[INSERT TEXT REGISTER]	[F4]	
[PRINT TEXT]	ESC - O	
[SET TEXT MARKER]	ESC - J	
[GOTO TEXT MARKER]	ESC - G	
[FORMAT PARAGRAPH]	ESC - F	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[CTRL-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE KEYBOARD LAYOUT FOR THE IBM 3101

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	0	(1 = YES, 0 = NO)
[HOME]	ESC - ALT-H	
[ZEND]	ESC - ALT-Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[ALT-A]	
[TAB CURSOR]	[ALT-F]	
[ZIP]	[ALT-G]	
[NEXT LINE]	[ALT-N]	
[LINE TOGGLE]	NOT USED	Type RETURN
[SCROLL UP]	[ALT-Q]	
[SCROLL DOWN]	[ALT-Z]	
[SCROLL RIGHT]	[ALT-Y]	
[SCROLL LEFT]	[ALT-T]	
[PREVIOUS WORD]	[ALT-K]	
[NEXT WORD]	[ALT-L]	
[PREVIOUS PARAGRAPH]	ESC - W	
[NEXT PARAGRAPH]	ESC - X	
[PAGE UP]	[ALT-W]	
[PAGE DOWN]	[ALT-X]	
[SCREEN TOGGLE]	[ALT-J]	
[BACKSPACE]	[<---]	
[DELETE]	[DEL]	
[ERASE TO END OF LINE]	[EOL]	Also called [EREOL] in manual.
[ERASE LINE]	[EOS]	
[DEL PREVIOUS WORD]	ESC - K	
[DEL NEXT WORD]	ESC - L	
[UNDO]	[ALT-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	ESC - Q	
[SET INSERT MODE]	NOT USED	Type RETURN
[RESET INSERT MODE]	NOT USED	Type RETURN
[SWITCH INSERT MODE]	[ALT-V]	
[REPEAT]	[ALT-R]	
[INDENT]	[ALT-P]	
[UNDENT]	[ALT-O]	
[FIND]	[PF 7]	
[REPLACE]	[PF 8]	
[CANCEL]	[ALT-C]	
[COPY TO TEXT REGISTER]	[PF 1]	
[MOVE TO TEXT REGISTER]	[PF 2]	
[INSERT TEXT REGISTER]	[PF 3]	
[PRINT TEXT]	[PF 4]	
[SET TEXT MARKER]	[PF 5]	
[GOTO TEXT MARKER]	[PF 6]	
[FORMAT PARAGRAPH]	ESC - O	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[ALT-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE KEYBOARD LAYOUT FOR THE TELEVIDEO 920C

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	CTRL-A	Used by function keys.
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)

[HOME]	ESC - CTRL-H	
[ZEND]	ESC - CTRL-Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[CTRL-S]	
[TAB CURSOR]	[CTRL-T]	Useful for fast cursor movement.
[ZIP]	[CTRL-G]	
[NEXT LINE]	[CTRL-N]	
[LINE TOGGLE]	NONE	Type RETURN
[SCROLL UP]	FUNCT - Up Arrow	
[SCROLL DOWN]	FUNCT - Down Arrow	
[SCROLL RIGHT]	FUNCT - Right Arrow	
[SCROLL LEFT]	FUNCT - Left Arrow	
[PREVIOUS WORD]	[CTRL-D]	
[NEXT WORD]	[CTRL-F]	
[PREVIOUS PARAGRAPH]	ESC - Q	
[NEXT PARAGRAPH]	ESC - Z	
[PAGE UP]	[CTRL-Q]	
[PAGE DOWN]	[CTRL-Z]	
[SCREEN TOGGLE]	[CTRL-Y]	
[BACKSPACE]	[CTRL-B]	
[DELETE]	[DEL]	
[ERASE TO END OF LINE]	ESC - X	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-V]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	TAB	
[NEXT CHAR LITERAL]	ESC - L	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[F10]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	[F1]	
[REPLACE]	[F2]	
[CANCEL]	[F3]	
[COPY TO TEXT REGISTER]	[F4]	
[MOVE TO TEXT REGISTER]	[F5]	
[INSERT TEXT REGISTER]	[F6]	
[PRINT TEXT]	[F7]	
[SET TEXT MARKER]	[F8]	
[GOTO TEXT MARKER]	[F9]	
[FORMAT PARAGRAPH]	[CTRL-W]	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[F11]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE KEYBOARD LAYOUT FOR THE TELEVIDEO 950

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	CTRL-A	Used by function keys.
2ND CHA2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)
[HOME]	ESC - CTRL-H	
[ZEND]	ESC - CTRL-Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[BACK TAB]	
[TAB CURSOR]	[CTRL-T]	Useful for fast cursor movement.
[ZIP]	[CLEAR SPACE]	
[NEXT LINE]	[LINE FEED]	
[LINE TOGGLE]	[CTRL-Y]	
[SCROLL UP]	[CTRL-E]	
[SCROLL DOWN]	[CTRL-C]	
[SCROLL RIGHT]	[CTRL-N]	
[SCROLL LEFT]	[CTRL-B]	
[PREVIOUS WORD]	[CTRL-D]	
[NEXT WORD]	[CTRL-F]	
[PREVIOUS PARAGRAPH]	ESC - E	
[NEXT PARAGRAPH]	ESC - C	
[PAGE UP]	[CTRL-W]	
[PAGE DOWN]	[CTRL-X]	
[SCREEN TOGGLE]	[HOME]	
[BACKSPACE]	[DEL]	
[DELETE]	[CHAR DELETE]	
[ERASE TO END OF LINE]	[LINE ERASE]	Also called [EREOL] in manual.
[ERASE LINE]	[LINE DELETE]	
[DEL PREVIOUS WORD]	[CTRL-S]	
[DEL NEXT WORD]	[CTRL-G]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	TAB	
[NEXT CHAR LITERAL]	ESC - L	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CHAR INSERT]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	[F1]	
[REPLACE]	[F2]	
[CANCEL]	[F3]	
[COPY TO TEXT REGISTER]	[F4]	
[MOVE TO TEXT REGISTER]	[F5]	
[INSERT TEXT REGISTER]	[F6]	
[PRINT TEXT]	[SEND]	
[SET TEXT MARKER]	[F7]	
[GOTO TEXT MARKER]	[F8]	
[FORMAT PARAGRAPH]	[PAGE ERASE]	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[F11]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

APPENDIX E
EXAMPLE KEYBOARD LAYOUTS
8086 VERSIONS
SPECIFIC SCREENS

DESCRIPTION OF FILES ON DISK (MSDOS and PCDOS)

The VEDIT files for the IBM Personal Computer and for the Tandy 2000 are distributed on the same diskettes. The following is a description of these files. YOU DO NOT have to perform the customization process, described in the manual, to produce a runnable version of VEDIT. You may use the pre-configured VEDIT supplied, which follows the "Example Keyboard Layout for the IBM Personal Computer" or the "Example Keyboard Layout for the Tandy 2000.

To configure a NEW version of VEDIT, follow the instructions in the VEDIT manual under Appendix A, (Customizing VEDIT). It is usually a good idea to use the pre-configured version and get the "feel" of VEDIT, before configuring your own version. Note that these versions of VEDIT are Memory Mapped versions which accesses the screen directly.

VEDSET.COM	The program used to perform customization. The manual describes the use of this program and the ".SET" files below.
VEDITPC.SET	File for producing the VEDIT version for the IBM PC.
VEDIT.COM	'Runnable' version of VEDIT for the IBM PC.
TANDY.SET	File for producing the VEDIT version for the Tandy 2000
TANDY.COM	'Runnable' version of VEDIT for the Tandy 2000.
VHELP.TXT	Help file for VEDIT. It should be copied to your System disk. You may edit this file for your needs.
PRINT.EXC	The Page Formatting macro. It should be copied to your System disk if you intend to use it.

DESCRIPTION OF FILES (CP/M-86)

The following is a brief description of the files currently supplied on diskette for the IBM Personal Computer. You will not have to perform the customization process, described in the manual, to produce a runnable version of VEDIT. You may use the pre-configured VEDIT.CMD file, which follows the "Example Keyboard Layout for the IBM Personal Computer". To customize a NEW version of VEDIT, follow the instructions in the VEDIT manual in Appendix A, (Customizing VEDIT).

Note that the diskette contains one version of VEDIT for IBM's or Digital Research's implementation of CP/M-86 and a second version of VEDIT for CompuView's implementation of CP/M-86. The latter have a "CPI" in the file name. Users of CompuView's CP/M-86 will need to use their "IBM-IN-B" program to read this VEDIT distribution disk.

VEDSET.CMD	The program used to perform customization. The manual describes the use of this program and the ".SET" file below.
VEDITPC.SET	File for producing the VEDIT version for the IBM PC.
VEDIT.CMD	'Runnable' version of VEDIT.
VEDSTCPI.CMD	The program used to perform customization for the "CompuView" CP/M-86 version of VEDIT.
VEDITCPI.SET	File for producing the "CompuView" CP/M-86 version of VEDIT for the IBM PC.
VEDITCPI.CMD	'Runnable' version of VEDIT as above.
VHELP.TXT	Help file for VEDIT. It should be copied to your System disk. You may edit this file for your needs.
PRINT.EXC	The Page Formatting macro. It should be copied to your System disk if you intend to use it.

EXAMPLE KEYBOARD LAYOUT FOR THE IBM PERSONAL COMPUTER (*)

"ESCAPE MODE CHARACTER #1"	NOT USED	Type [RETURN]
[HOME]	[CTRL-HOME]	
[ZEND]	[CTRL-END]	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[F3]	
[TAB CURSOR]	[F4]	Useful for fast cursor movement.
[ZIP]	[F6]	
[NEXT LINE]	[F5]	
[LINE TOGGLE]	[CTRL-J]	
[SCROLL UP]	[HOME]	
[SCROLL DOWN]	[END]	
[SCROLL RIGHT]	[F2]	
[SCROLL LEFT]	[F1]	
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	[CTRL-Pg Up]	
[NEXT PARAGRAPH]	[CTRL-Pg Dn]	
[PAGE UP]	[Pg Up]	
[PAGE DOWN]	[Pg Dn]	
[SCREEN TOGGLE]	[CTRL-K]	
[BACKSPACE]	[<---]	
[DELETE]	[Del]	
[ERASE TO END OF LINE]	[CTRL-Z]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-N]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	[CTRL-L]	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[Ins]	
[REPEAT]	[CTRL-R]	
[INDENT]	[F8]	
[UNDENT]	[F7]	
[FIND]	[ALT-F1]	
[REPLACE]	[ALT-F2]	
[CANCEL]	[ALT-F3]	
[COPY TO TEXT REGISTER]	[F9]	
[MOVE TO TEXT REGISTER]	[ALT-F9]	
[INSERT TEXT REGISTER]	[F10]	
[PRINT TEXT]	[CTRL-P]	
[SET TEXT MARKER]	[CTRL-S]	
[GOTO TEXT MARKER]	[CTRL-G]	
[FORMAT PARAGRAPH]	[CTRL-F]	
[VISUAL ESCAPE]	[ESC]	
[VISUAL EXIT]	[CTRL-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

(*) Except Concurrent CP/M-86 - see following page

EXAMPLE KEYBOARD LAYOUT FOR THE IBM PC WITH CONCURRENT CP/M-86

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	?	
"COMMON 2ND CHARACTER #2..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)
[HOME]	ESC - H	
[ZEND]	ESC - Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[F3]	
[TAB CURSOR]	[F4]	Useful for fast cursor movement.
[ZIP]	[F6]	
[NEXT LINE]	[F5]	
[LINE TOGGLE]	[CTRL-T]	
[SCROLL UP]	[CTRL-Q]	
[SCROLL DOWN]	[CTRL-A]	
[SCROLL RIGHT]	[F2]	
[SCROLL LEFT]	[F1]	
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	[ESC - W]	
[NEXT PARAGRAPH]	[ESC - X]	
[PAGE UP]	[Pg Up]	
[PAGE DOWN]	[Pg Dn]	
[SCREEN TOGGLE]	[CTRL-Y]	
[BACKSPACE]	[<---]	
[DELETE]	[Del]	
[ERASE TO END OF LINE]	[CTRL-Z]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-N]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	[ESC - Q]	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[Ins]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	[CTRL-J]	
[REPLACE]	[CTRL-K]	
[CANCEL]	[CTRL-L]	
[COPY TO TEXT REGISTER]	[F9]	
[MOVE TO TEXT REGISTER]	[F10]	
[INSERT TEXT REGISTER]	[F8]	
[PRINT TEXT]	[F7]	
[SET TEXT MARKER]	[ESC - S]	
[GOTO TEXT MARKER]	[ESC - G]	
[FORMAT PARAGRAPH]	[CTRL-F]	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[CTRL-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE CUSTOMIZATION FOR IBM PC VERSION 1.16

The customization session used to create the pre-configured VEDIT is listed below.

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (AD) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C
     HEX CODE FOR CURSOR CHARACTER                (5F) -- 5F
     HEX CODE FOR CLEAR SCREEN CHARACTER          (20) -- 20
     HEX CODE FOR STATUS LINE CHARACTER           (2D) -- 2D
     HEX CODE FOR TAB EXPAND CHARACTER            (20) -- 20

4.)  EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

     CURSOR TYPE (0=UNDERLINE, 1=BLINK BLOCK, 2=BLOCK, 3=ATTRIBUTE) -- 3
     CURSOR BLINK RATE, SMALL # IS FAST (2MHZ - 20, 4MHZ - 40) -- 40
     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERT CHARACTER (59) -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP (0=OFF)                   -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 3) -- 3
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 24  (Use 25 for PCDOS)
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER SCREEN ATTRIBUTE CODE FOR NON-REVERSE VIDEO IN HEX (07) -- 07
     ENTER SCREEN ATTRIBUTE CODE FOR REVERSE VIDEO IN HEX     (70) -- 70
     ENTER SCREEN ATTRIBUTE CODE FOR CURSOR IN HEX            (F0) -- F0

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 6144
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 12

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

EXAMPLE KEYBOARD LAYOUT FOR THE TANDY 2000

"ESCAPE MODE CHARACTER #1"	NOT USED	Type [RETURN]
[HOME]	[CTRL- HOME]	
[ZEND]	[CTRL- END]	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[F9]	
[TAB CURSOR]	[F10]	Useful for fast cursor movement.
[ZIP]	[0]	On Keypad
[NEXT LINE]	[END]	
[LINE TOGGLE]	[CTRL-0]	"0" on keypad
[SCROLL UP]	[CTRL- Up Arrow]	
[SCROLL DOWN]	[CTRL- Down Arrow]	
[SCROLL RIGHT]	[CTRL- Right Arrow]	
[SCROLL LEFT]	[CTRL- Left Arrow]	
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	[CTRL-Pg Up]	
[NEXT PARAGRAPH]	[CTRL-Pg Dn]	
[PAGE UP]	[Pg Up]	
[PAGE DOWN]	[Pg Dn]	
[SCREEN TOGGLE]	[HOME]	
[BACKSPACE]	[<---]	
[DELETE]	[DELETE]	
[ERASE TO END OF LINE]	[CTRL-Z]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-N]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	[CTRL-L]	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[Ins]	
[REPEAT]	[CTRL-R]	
[INDENT]	[F12]	
[UNDENT]	[F11]	
[FIND]	[F1]	
[REPLACE]	[F2]	
[CANCEL]	[F3]	
[COPY TO TEXT REGISTER]	[F4]	
[MOVE TO TEXT REGISTER]	[F5]	
[INSERT TEXT REGISTER]	[F6]	
[PRINT TEXT]	[PRINT]	Same as CTRL-P
[SET TEXT MARKER]	[F7]	
[GOTO TEXT MARKER]	[F8]	
[FORMAT PARAGRAPH]	[CTRL-F]	
[VISUAL ESCAPE]	[BREAK]	
[VISUAL EXIT]	[ESC]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE CUSTOMIZATION FOR TANDY 2000 VERSION 1.16

The customization session used to create the pre-configured VEDIT is listed below. BE SURE TO ENTER "5.)" VALUES EXACTLY AS SHOWN

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (AD) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C
     HEX CODE FOR CURSOR CHARACTER                (5F) -- 5F
     HEX CODE FOR CLEAR SCREEN CHARACTER          (20) -- 20
     HEX CODE FOR STATUS LINE CHARACTER           (2D) -- 2D
     HEX CODE FOR TAB EXPAND CHARACTER            (20) -- 20

4.)  EXPAND TAB WITH SPACES         (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE           (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT    (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT    (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE   (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS           (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS    (0=NO, 1=YES) -- 0

     CURSOR TYPE (0=UNDERLINE, 1=BLINK BLOCK, 2=BLOCK, 3=ATTRIBUTE) -- 1
     CURSOR BLINK RATE, SMALL # IS FAST (2MHZ - 20, 4MHZ - 40) -- 80
     INDENT INCREMENT  (1 - 20, SUGGEST 4)                  -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)        -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERT CHARACTER (59) -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP (0=OFF)                     -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 3) -- 3
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)           -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)          -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                  -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 25
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER SCREEN ATTRIBUTE CODE FOR NON-REVERSE VIDEO IN HEX (07) -- 08
     ENTER SCREEN ATTRIBUTE CODE FOR REVERSE VIDEO IN HEX     (70) -- 88
     ENTER SCREEN ATTRIBUTE CODE FOR CURSOR IN HEX            (F0) -- 28

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 6144
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 12

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 1
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES        -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)             -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

EXAMPLE KEYBOARD LAYOUT FOR THE Z100

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	[?]	
"COMMON 2ND CHARACTER #2..."	[0]	
"UPPER/LOWER CASE ESCAPE..."	0	(1 = YES, 0 = NO)

[HOME]	ESC - h	
[ZEND]	ESC - z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[7]	on keypad
[TAB CURSOR]	[8]	Useful for fast cursor movement.
[ZIP]	[9]	on keypad
[NEXT LINE]	[LINE FEED]	
[LINE TOGGLE]	[-]	on keypad
[SCROLL UP]	[6]	on keypad
[SCROLL DOWN]	[3]	on keypad
[SCROLL RIGHT]	[.]	on keypad
[SCROLL LEFT]	[0]	on keypad
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	[5]	on keypad
[NEXT PARAGRAPH]	[2]	on keypad
[PAGE UP]	[4]	on keypad
[PAGE DOWN]	[1]	on keypad
[SCREEN TOGGLE]	[CTRL-S]	
[BACKSPACE]	[BACK SPACE]	
[DELETE]	[DELETE]	
[ERASE TO END OF LINE]	[CTRL-Z]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-N]	
[UNDO]	[F12]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	[CTRL-L]	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[ENTER]	on keypad
[REPEAT]	[F0]	
[INDENT]	[F11]	
[UNDENT]	[F10]	
[FIND]	[F1]	
[REPLACE]	[F2]	
[CANCEL]	[F3]	
[COPY TO TEXT REGISTER]	[F4]	
[MOVE TO TEXT REGISTER]	[F5]	
[INSERT TEXT REGISTER]	[F6]	
[PRINT TEXT]	[F7]	
[SET TEXT MARKER]	[F8]	
[GOTO TEXT MARKER]	[F9]	
[FORMAT PARAGRAPH]	[INS LINE]	
[VISUAL ESCAPE]	[ESC-ESC]	
[VISUAL EXIT]	[HELP]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE CRT CUSTOMIZATION FOR THE Z100

The customization session used to create the pre-configured VEDIT is listed below. For clarity sake, each reply below is preceded by "--", which does not appear is the actual customization.

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C

4.)  EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 2
     BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 1
     EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERSION CHAR. (59)   -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP (0=OFF)                   -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1)  -- 1
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 25
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 21
     ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 80

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 6144
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 12

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     ENTER DECIMAL VALUE  (4MHZ = 76, 2MHZ = 38) -- 76
     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

VEDIT Notes for IBM Displaywriter

In order for you to bring up VEDIT as easily as possible on your IBM Displaywriter, we have included a pre-configured "ready to run" version for this machine in the disk file "VEDIT.CMD". You therefore DO NOT need to perform the customization in order to first begin using VEDIT. You will probably wish to perform the customization later, in order to change the keyboard layout or change some of the default parameters. Attached to these notes should be "Example Keyboard Layout for IBM Displaywriter" and "Example Customization for IBM Displaywriter".

The keyboard layout describes the keyboard strokes needed to perform the VEDIT editing functions. Note two differences between the Displaywriter keyboard and most other keyboards as described in the VEDIT manual. One is that the Displaywriter uses the "CODE" key instead of the more common "CTRL" key. Second, the Displaywriter does not have an "ESC" key and therefore we have adopted the "INDEX" key to perform the same function with VEDIT. (Only the customization needs the true ESC character, which is produced by typing the upper-most right hand key in the left keypad, or alternately by typing the "1/4" key while holding the CODE and SHIFT keys down).

You don't need to refer to the "Example Customization for IBM Displaywriter" until you customize a new version of VEDIT. When customizing the VEDIT parameters you should generally only change those parameters described in tasks 4.) and 6.) of appendix A of the manual. Answer the other questions with the values in the "Example Customization for IBM Displaywriter". You may change the keyboard layout any way you wish. When selecting the CRT terminal type at the beginning of the customization, note that there is an entry for the Displaywriter; DON'T select the IBM 3101. Please give us a call if you need any assistance.

Before you begin using VEDIT, you should copy our distribution disk onto your own disk. This is easy if your disk in drive "A" has at least 230K of space free. Simply insert our disk in drive "B", type "CODE - C" and then use the "PIP" program to copy all files to the "A" drive. Remove our disk and place an empty formatted disk into drive "B". Type "CODE - C" twice and use PIP again to copy our files to your own disk. You can then erase all but "VEDIT.CMD" from your disk in drive "A". The disk you have just made in drive "B" should be used when you customize a new VEDIT. The file "VEDIT.CMD" is all you need to run VEDIT.

The other files on our distribution disk are "VDSETCRT.CMD", "VEDIT86.SET" and "CRT.TBL" which are needed for the customization. "CHECKSUM.DOC" explains how to remove a checksum error, and "AUTOLOAD.DOC" contains a discussion and example of the autoload feature of VEDIT.

EXAMPLE KEYBOARD LAYOUT FOR IBM DISPLAYWRITER

Functions using the INDEX key require first typing the INDEX key and then the second key. The [ESC] key is the upper right hand key in the left keypad.

"ESCAPE MODE CHARACTER #1"	[INDEX]	
"ESCAPE MODE CHARACTER #2"	[ESC]	Top right key in left keypad
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	

[HOME]	INDEX - Y	
[ZEND]	INDEX - Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[FIND]	
[TAB CURSOR]	[CODE-G]	Useful for fast cursor movement.
[ZIP]	[GOTO]	
[NEXT LINE]	[CODE-N]	
[LINE TOGGLE]	[CODE-L]	
[SCROLL UP]	[CODE-W]	
[SCROLL DOWN]	[CODE-S]	
[SCROLL RIGHT]	[CODE-Y]	
[SCROLL LEFT]	[CODE-T]	
[PREVIOUS WORD]	[CODE-D]	
[NEXT WORD]	[CODE-F]	
[PREVIOUS PARAGRAPH]	[CODE-Q]	
[NEXT PARAGRAPH]	[CODE-A]	
[PAGE UP]	Unmarked key right of [FIND]	
[PAGE DOWN]	Unmarked key right of [GOTO]	
[SCREEN TOGGLE]	[CODE-K]	
[BACKSPACE]	[BKSP]	
[DELETE]	[DEL]	
[ERASE TO END OF LINE]	[CODE-Z]	Also called [EREOL] in manual.
[ERASE LINE]	[CODE-X]	
[DEL PREVIOUS WORD]	[CODE-C]	
[DEL NEXT WORD]	[CODE-V]	
[UNDO]	[CODE-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	INDEX - L	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CODE-B]	
[REPEAT]	[CODE-R]	
[INDENT]	[CODE-P]	
[UNDENT]	[CODE-O]	
[FIND]	INDEX - 1	
[REPLACE]	INDEX - 2	
[CANCEL]	INDEX - 3	
[COPY TO TEXT REGISTER]	INDEX - 4	
[MOVE TO TEXT REGISTER]	INDEX - 5	
[INSERT TEXT REGISTER]	INDEX - 6	
[PRINT TEXT]	INDEX - 7	
[SET TEXT MARKER]	INDEX - S	
[GOTO TEXT MARKER]	INDEX - G	
[FORMAT PARAGRAPH]	INDEX - F	
[VISUAL ESCAPE]	INDEX - INDEX	
[VISUAL EXIT]	[CODE-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE CUSTOMIZATION FOR IBM DISPLAYWRITER VERSION 1.38

The customization session used to create the pre-configured VEDIT is listed below. Task 1 of the customization is the CRT selection from the menu, for which the Displaywriter entry should be selected. The pre-configured keyboard layout is described by the enclosed layout sheet. For clarity sake, each reply below is preceded by "--", which does not appear is the actual customization.

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 0A
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C

4.)  EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERT CHARACTER (59) -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP (0=OFF)                   -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1) -- 1
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 24
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 80

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 8192
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 12

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     ENTER DECIMAL VALUE  (4MHZ = 76, 2MHZ = 38) -- 76
     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

EXAMPLE KEYBOARD LAYOUT FOR THE NEC APC

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	0	The capital letter "0"
"COMMON 2ND CHARACTER #2..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)

[HOME]	ESC - H	
[ZEND]	ESC - Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[PF 9]	
[TAB CURSOR]	[PF 10]	
[ZIP]	[PF 11]	
[NEXT LINE]	[CTRL-R]	
[LINE TOGGLE]	[CTRL-G]	
[SCROLL UP]	[CTRL-E]	
[SCROLL DOWN]	[CTRL-D]	
[SCROLL RIGHT]	[PF 8]	
[SCROLL LEFT]	[PF 7]	
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	[CTRL-Q]	
[NEXT PARAGRAPH]	[CTRL-A]	
[PAGE UP]	[PF 5]	
[PAGE DOWN]	[PF 6]	
[SCREEN TOGGLE]	[CLEAR HOME]	
[BACKSPACE]	[BACK SPACE]	
[DELETE]	[DEL]	
[ERASE TO END OF LINE]	[PF 14]	Also called [EREOL] in manual.
[ERASE LINE]	[PF 15]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-N]	
[UNDO]	[PF 16]	Last Black key
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	ESC - L	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[INS]	
[REPEAT]	[PF 1]	First Black key
[INDENT]	[PF 13]	
[UNDENT]	[PF 12]	
[FIND]	[PF 2]	
[REPLACE]	[PF 3]	
[CANCEL]	[PF 4]	
[COPY TO TEXT REGISTER]	[PF 18]	Second Grey Key
[MOVE TO TEXT REGISTER]	[PF 19]	Note: The first grey key "PF17"
[INSERT TEXT REGISTER]	[PF 20]	cannot be used with VEDIT
[PRINT TEXT]	[PRINT]	
[SET TEXT MARKER]	[PF 21]	
[GOTO TEXT MARKER]	[PF 22]	
[FORMAT PARAGRAPH]	[CTRL-F]	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[BREAK]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

Note: VEDIT must be customized for 25 screen lines.

VEDIT Notes for VICTOR 9000

In order for you to start using VEDIT as quickly as possible on your Victor 9000, we have included a pre-configured "ready to run" version for this machine in the disk file "VEDIT.CMD". You therefore DO NOT need to perform the customization in order to begin using VEDIT. You will probably wish to perform the customization later, in order to change the keyboard layout or change some of the default parameters. The following page "Example Keyboard Layout for Victor 9000" gives the keyboard layout used in the pre-configured version.

The only unusual aspect of the Victor 9000 keyboard is that the ESC key requires use of the ALT key. This is of no significance in the visual mode, but is somewhat inconvenient in the command mode, since some commands require an ESC to terminate text strings. each command line ends in two ESC. You may, therefore, find it more convenient to use "Explicit String Delimiters", in which case the ESC key will be rarely used.

When customizing VEDIT, you must select the VICTOR 9000 from the menu of supported terminals. The attached sheet "Example Customization for Victor 9000" shows the customization values used for the pre-configured version.

Before you begin using VEDIT, you should copy our distribution disk onto your own disk. This is best done with the copy program "DCOPY.CMD". To use VEDIT, you only need the file "VEDIT.CMD" (either the one we supplied, or one you customized), which you can simply copy to your system disk on drive "A". You probably will also want the help file "VHELP.TXT" on drive "A".

The other files on our distribution disk are "VDSETCRT.CMD", "VEDIT86.SET" and "CRT.TBL" which are needed for the customization. "CHECKSUM.DOC" explains how to remove a checksum error, and "AUTOLOAD.DOC" contains a discussion and example of the autoload feature of VEDIT.

EXAMPLE KEYBOARD LAYOUT FOR VICTOR 9000

Note that typing the [ESC] key requires holding down the ALT key.

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	0	
[HOME]	ESC - h	
[ZEND]	ESC - z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[ALT-TAB]	
[TAB CURSOR]	[ALT-F]	Useful for fast cursor movement.
[ZIP]	[REQ CAN]	
[NEXT LINE]	[OFF UNDL]	
[LINE TOGGLE]	[ALT-A]	
[SCROLL UP]	[ALT-SCRL]	
[SCROLL DOWN]	[SCRL]	
[SCROLL RIGHT]	[ALT-Y]	
[SCROLL LEFT]	[ALT-T]	
[PREVIOUS WORD]	[WORD <--]	
[NEXT WORD]	[WORD -->]	
[PREVIOUS PARAGRAPH]	[ALT-Q]	
[NEXT PARAGRAPH]	[ALT-Z]	
[PAGE UP]	[ALT-W]	
[PAGE DOWN]	[ALT-X]	
[SCREEN TOGGLE]	[SHIFT - CLR HOME]	
[BACKSPACE]	[BACKSPACE]	
[DELETE]	[DEL]	
[ERASE TO END OF LINE]	[ERASE EOL]	
[ERASE LINE]	[ALT- ERASE EOL]	
[DEL PREVIOUS WORD]	[ALT-C]	
[DEL NEXT WORD]	[ALT-V]	
[UNDO]	[ALT-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	[ALT-L]	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[INS]	
[REPEAT]	[ALT-R]	
[INDENT]	[ALT-P]	
[UNDENT]	[ALT-O]	
[FIND]	FUNCTION KEY [1]	
[REPLACE]	FUNCTION KEY [2]	
[CANCEL]	[PAUSE CONT]	Same as [CTRL-S]
[COPY TO TEXT REGISTER]	FUNCTION KEY [3]	
[MOVE TO TEXT REGISTER]	FUNCTION KEY [4]	
[INSERT TEXT REGISTER]	FUNCTION KEY [5]	
[PRINT TEXT]	FUNCTION KEY [8]	
[SET TEXT MARKER]	FUNCTION KEY [6]	
[GOTO TEXT MARKER]	FUNCTION KEY [7]	
[FORMAT PARAGRAPH]	FUNCTION KEY [9]	
[VISUAL ESCAPE]	[ALT-E]	
[VISUAL EXIT]	FUNCTION KEY [10]	
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE CUSTOMIZATION FOR VICTOR 9000

The customization session used to create the pre-configured VEDIT is listed below. Task 0 is the CRT selection from the menu, for which the Heath H19 entry should be selected. For clarity sake, each reply below is preceded by "--", which does not appear is the actual customization.

```
0.)  SELECT CRT TERMINAL -- 46   (Victor 9000)

3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C

4.)  EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERT CHARACTER (59) -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP (0=OFF)                   -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1) -- 3
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL    -- 24
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 80

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ      -- 4096
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 12

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     ENTER DECIMAL VALUE  (4MHZ = 76, 2MHZ = 38) -- 76
     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

EXAMPLE KEYBOARD LAYOUT FOR THE TI PROFESSIONAL COMPUTER

"ESCAPE MODE CHARACTER #1"	NOT USED	Type [RETURN]
[HOME]	[CTRL- F11]	
[ZEND]	[CTRL- F12]	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[SHIFT-TAB]	
[TAB CURSOR]	[CTRL-T]	Useful for fast cursor movement.
[ZIP]	[HOME]	
[NEXT LINE]	[LINE FEED]	
[LINE TOGGLE]	[CTRL-L]	
[SCROLL UP]	[CTRL- Up Arrow]	
[SCROLL DOWN]	[CTRL- Down Arrow]	
[SCROLL RIGHT]	[CTRL- Right Arrow]	
[SCROLL LEFT]	[CTRL- Left Arrow]	
[PREVIOUS WORD]	[CTRL-V]	
[NEXT WORD]	[CTRL-B]	
[PREVIOUS PARAGRAPH]	[CTRL- F5]	
[NEXT PARAGRAPH]	[CTRL- F6]	
[PAGE UP]	[F5]	
[PAGE DOWN]	[F6]	
[SCREEN TOGGLE]	[CTRL-K]	
[BACKSPACE]	[<---]	
[DELETE]	[Del]	
[ERASE TO END OF LINE]	[CTRL-Z]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	[CTRL-C]	
[DEL NEXT WORD]	[CTRL-N]	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	[CTRL-Q]	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[Ins]	
[REPEAT]	[F4]	
[INDENT]	[F8]	
[UNDENT]	[F7]	
[FIND]	[F1]	
[REPLACE]	[F2]	
[CANCEL]	[F3]	
[COPY TO TEXT REGISTER]	[F9]	
[MOVE TO TEXT REGISTER]	[CTRL- F9]	
[INSERT TEXT REGISTER]	[F10]	
[PRINT TEXT]	[PRNT]	
[SET TEXT MARKER]	[F11]	
[GOTO TEXT MARKER]	[F12]	
[FORMAT PARAGRAPH]	[CTRL-F]	
[VISUAL ESCAPE]	ESC	
[VISUAL EXIT]	[CTRL-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

Note: For MSDOS choose "DEC VT-100" in CRT menu and "# screen lines" = 25
For CP/M-86 choose "H19" in CRT menu and "# screen lines" = 24

APPENDIX F
EXAMPLE KEYBOARD LAYOUTS
8080/Z80 VERSIONS
MEMORY MAPPED SCREENS

VEDIT Notes for TRS-80 Model I

The diskette supplied with the TRS-80 Model I version of VEDIT contains two configurations of the software, one for the standard Model I with a CP/M base (or ORG) at 4200 Hex and the second for the modified Model I with a CP/M base at 0000 Hex. Modifications such as the Omicron Mapper use a CP/M base at 0000.

In order to let you to use VEDIT as quickly as possible, we have including a "ready to run" VEDIT on the disk, which does not need to customized. This VEDIT follows the "Example Keyboard Layout for the TRS-80 Model I" and the "Example Customization for TRS-80 Model I". The two files are:

VEDIT@00.COM Ready to run VEDIT for ORG 0000 Systems

VEDIT@42.COM Ready to Run VEDIT for ORG 4200 Systems

When you customize your own VEDIT, the files VEDST@00.COM and VTZM@00.SET are for use with a CP/M base of 0000, while the files VEDST@42.COM and VTZM@42.SET are for use with a CP/M base of 4200 Hex. Therefore, the CP/M command to invoke the customization program for the CP/M at 0000 is:

```
VEDST@00 VTZM@00 VEDIT
```

The command to invoke the customization program for the CP/M at 4200 is:

```
VEDST@42 VTZM@42 VEDIT
```

We recommend using the Default Keyboard Layout. However, since the TRS-80 Model I does not have an ESCAPE key ([ESC] in the manual), another key must be used for the ESCAPE key in both command and visual modes. We recommend the use of the Up Arrow ([CTRL-Z]) on the Model I.

Also, producing control characters from the keyboard requires holding both the SHIFT and Down Arrow, and then typing the letter.

Please note that VEDIT has a software switch which will perform any necessary reversal of upper and lower case letters typed on the keyboard. The Model I keyboard normally produces upper case letters, and with the shift key depressed, it produces lower case letters. Some of the CP/M implementations for the Model I will reverse these keyboard characters, while others will not. If your CP/M does not reverse the letters, you will probably want VEDIT to do so. This reversal may be specified during Task 4. of the customization, or with switch 7 of the "ES" command while running VEDIT.

Customization Notes for TRS-80 Model I

The TRS-80 user running the customization program should reference these notes to help answer some of the questions. The numbers below refer to the customization Tasks described in Appendix A of the manual. An example customization follows. Since the enclosed keyboard layout covers Task 1, these notes cover only Tasks 3 through 6. Users having at least a little experience with VEDIT can try other values, unless we indicate that a specific value must be used.

3.2.) Use "0A" for the Command mode escape character in order to make the Down Arrow the Escape key.

3.5) Use "7F" for the cursor character.

4.5) Use "1" to reverse upper and lower case letters.

5.1) Must use value of "16".

5.2) A value of around "10-14" is suggested. Maximum is "15".

5.3) Personal preference. A value of "3" is reasonable.

5.4) Personal preference. A value of "12" is reasonable.

5.5) Must use value of "64".

5.6) Normally use value of "64".

5.7) Must use value of "3C00", (or FC00 with Shuffleboard).

5.8) Must use value of "0".

EXAMPLE CUSTOMIZATION FOR TRS-80 MODEL I

For clarity sake, each reply below is preceded by "--" which does not appear in the actual customization. Please note that the Screen address is "3C00" for an unmodified machine, and "FC00" with the Shuffleboard modification.

```
3.) HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- 2D
    HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 0A
    HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
    HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
    HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C
    HEX CODE FOR CURSOR CHARACTER (5F)       -- 7F
    HEX CODE FOR SCREEN CLEAR CHARACTER (20) -- 20
    HEX CODE FOR STATUS LINE CHARACTER (2D)  -- 2D
    HEX CODE FOR TAB EXPAND CHARACTER (20)   -- 20

4.) EXPAND TAB WITH SPACES         (0=NO, 1=YES) -- 0
    AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
    BEGIN IN VISUAL MODE           (0=NO, 1=YES) -- 1
    POINT PAST TEXT REG. INSERT    (0=NO, 1=YES) -- 1
    IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
    CLEAR SCREEN ON VISUAL EXIT    (0=NO, 1=YES) -- 0
    REVERSE UPPER AND LOWER CASE   (0=NO, 1=YES) -- 1
    IGNORE SEARCH ERRORS           (0=NO, 1=YES) -- 0
    EXPLICIT STRING TERMINATORS    (0=NO, 1=YES) -- 0

    CURSOR TYPE (0, 1, 2) -- 0
    CURSOR BLINK RATE, SMALL # IS FAST (2MHZ - 20, 4MHZ - 40) -- 18
    INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
    LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
    DECIMAL CODE FOR CONDITIONAL CONVERSION CHAR. (59 )  -- 59
    LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
    RIGHT MARGIN FOR WORD WRAP IN DECIMAL (0=OFF)        -- 0
    HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1) -- 1
    CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
    VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
    HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.) ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 16
    ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 12
    ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
    ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 12
    ENTER SCREEN LINE LENGTH IN DECIMAL       -- 64
    ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 64
    ENTER ADDRESS OF SCREEN IN HEXADECIMAL    -- 3C00
    ENTER NUMBER OF VIDEO BOARD INITIALIZATION BYTES -- 0

6.) SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 4096
    SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 8

    DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y
    BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0
    USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
    ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
    SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 0
    REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 0
```

VEDIT

DESCRIPTION OF FILES FOR THE TRS-80 MODEL II

The following is a brief description of the files currently supplied on diskette for the TRS-80 Model II versions. In order to let you use VEDIT as quickly as possible, two "ready to run" VEDIT files are supplied, which do not need to be customized. These files follow the "Example Keyboard Layout for TRS-80 Model II" and the "Customization Notes for TRS-80 Model II".

VEDSET.COM — The program used to perform the customization. The manual describes the use of this program and the ".SET" files below.

VEDIT2P.COM — The ready to run version of VEDIT for the Pickles & Trout CP/M.

VEDIT2G.COM — The ready to run version of VEDIT for all other CP/M TRS-80 Model II systems.

VEDIT2P.SET — File for producing the VEDIT version for the Pickles & Trout CP/M.

VEDIT2G.SET — File for producing the VEDIT version for all other CP/M, TRS-80 Model II systems.

Customization Notes for TRS-80 Model II

Users running the customization program may wish to use these notes to help answer some of the questions. In fact, some necessary information is only contained here, and not in any of the Radio Shack supplied documentation. An example customization follows. Since the enclosed keyboard layouts give examples for Task 1, these notes describe only Tasks 3 through 6. Users having a little experience with VEDIT can try other values, unless we indicate that a specific value must be used.

3.) All suggested values are used, but you may use "AD" instead of "2D" for the continuation character, since reverse video is available.

4.10) Use cursor types 0, 1 or 2, depending upon personal preference. Types 1 and 2 seem most appropriate.

5.1) Must use value of "24".

5.2) A value around 19 - 21 is suggested. Maximum is "23".

5.3) Personal preference. A value of "4" is reasonable.

5.4) Personal preference. A value of "20" is reasonable.

5.5) Must use value of "80".

5.6 Normally use value of "80".

5.7 Must use value of "F800".

5.8 Must use value of "0".

EXAMPLE CUSTOMIZATION FOR TRS-80 MODEL II

For clarity sake, each reply below is preceded by "--", which does not appear is the actual customization.

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C
     HEX CODE FOR CURSOR CHARACTER       (5F) -- 5F
     HEX CODE FOR SCREEN CLEAR CHARACTER (20) -- 20
     HEX CODE FOR STATUS LINE CHARACTER  (2D) -- 2D
     HEX CODE FOR TAB EXPAND CHARACTER   (20) -- 20

4.)  EXPAND TAB WITH SPACES         (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE           (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT    (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT    (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE   (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS           (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS    (0=NO, 1=YES) -- 0

     CURSOR TYPE (0, 1, 2) -- 1
     CURSOR BLINK RATE, SMALL # IS FAST (2MHZ - 20, 4MHZ - 40) -- 40
     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERSION CHAR. (59 ) -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP IN DECIMAL (0=OFF)        -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1) -- 1
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 24
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 4
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER SCREEN LINE LENGTH IN DECIMAL       -- 80
     ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 80
     ENTER ADDRESS OF SCREEN IN HEXADECIMAL    -- F800
     ENTER NUMBER OF VIDEO BOARD INITIALIZATION BYTES -- 0

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 6144
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 12

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

EXAMPLE KEYBOARD LAYOUT FOR TRS-80 MODEL II

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	
[HOME]	ESC - H	
[ZEND]	ESC - Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[BREAK]	
[TAB CURSOR]	[CTRL-F]	Useful for fast cursor movement.
[ZIP]	[HOLD]	
[NEXT LINE]	[CTRL-N]	
[LINE TOGGLE]	[CTRL-G]	
[SCROLL UP]	[CTRL-Q]	
[SCROLL DOWN]	[CTRL-Z]	
[SCROLL RIGHT]	[CTRL-Y]	
[SCROLL LEFT]	[CTRL-T]	
[PREVIOUS WORD]	[CTRL-J]	
[NEXT WORD]	[CTRL-K]	
[PREVIOUS PARAGRAPH]	ESC - F1	
[NEXT PARAGRAPH]	ESC - F2	
[PAGE UP]	[F1]	
[PAGE DOWN]	[F2]	
[SCREEN TOGGLE]	[CTRL-W]	
[BACKSPACE]	[BACK SPACE]	
[DELETE]	[CTRL-D]	
[ERASE TO END OF LINE]	[CTRL-L]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	ESC - J	
[DEL NEXT WORD]	ESC - K	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[TAB]	
[NEXT CHAR LITERAL]	ESC - Q	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CTRL-V]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	ESC - F	
[REPLACE]	ESC - R	
[CANCEL]	ESC - 0	
[COPY TO TEXT REGISTER]	ESC - C	
[MOVE TO TEXT REGISTER]	ESC - M	
[INSERT TEXT REGISTER]	ESC - I	
[PRINT TEXT]	ESC - P	
[SET TEXT MARKER]	ESC - S	
[GOTO TEXT MARKER]	ESC - G	
[FORMAT PARAGRAPH]	ESC - B	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[CTRL-E]	Normally used exit to command mode
[RESTART EDITOR]	NOT USED	Use "EA" Command.

SUGGESTED CUSTOMIZATION FOR IMSAI VIOC

A suggested IMSAI VIOC customization session is listed below. The values depending upon memory size are based on a 40K system.

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- AD
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 5B
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 5D
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C
     HEX CODE FOR CURSOR CHARACTER (5F)        -- 5F
     HEX CODE FOR SCREEN CLEAR CHARACTER (20) -- 20
     HEX CODE FOR STATUS LINE CHARACTER (2D)  -- 2D
     HEX CODE FOR TAB EXPAND CHARACTER (20)   -- 20

4.)  EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
     AUTO-BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

     CURSOR TYPE (0, 1, 2, 3) -- 1
     CURSOR BLINK RATE, SMALL # IS FAST (2MHZ - 20, 4MHZ - 40) -- 20
     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERSION CHAR. (59)   -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP IN DECIMAL (0=OFF)        -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1) -- 1
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL   -- 24
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 4
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER SCREEN LINE LENGTH IN DECIMAL       -- 80
     ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 80
     ENTER ADDRESS OF SCREEN IN HEXADECIMAL    -- F800
     ENTER NUMBER OF VIDEO BOARD INITIALIZATION BYTES -- 0

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 4096
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 8

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y

     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 1
```

APPENDIX G
EXAMPLE KEYBOARD LAYOUTS
8080/Z80 VERSIONS
SPECIFIC SCREENS

Notes for SuperBrain Version of VEDIT

The diskette is supplied as either single density "SD/128" or double density "DD/512". (Unless specified otherwise, we supply double density.) Users running double density, and receiving a single density diskette, can use the SuperBrain supplied CP/M system to operate drive A at double density and drive B at single density. In this way the files may be copied to a double density diskette.

The following notes are applicable to the customization process:

With some versions of the SuperBrain DOS, <CTRL-W> cannot be used, because it is intercepted and will lock up the machine. In this case you can change [CURSOR RIGHT] to be <CTRL-F>, not implement [TAB CURSOR], and use <CTRL-D> for [PAGE UP]. Otherwise, the DEFAULT KEYBOARD LAYOUT is usable.

The cursor control keys are also intercepted very early by the Superbrain's BIOS (DOS 3.0) and can only be used if you have a modified BIOS which does not intercept the cursor keys, nor strips the high order bit produced by the cursor and keypad keys.

If you would like to use the cursor control keys and the numeric keypad, you can install the patch on the following pages to your CP/M operating system.

SuperBrain Patch for use with VEDIT

For SuperBrain DOS version 3.0

This patch allows the cursor keys and the numeric keypad to be used for editing functions with VEDIT. The patch is made to the BIOS section of the CP/M operating system and only applies to DOS version 3.0. Later versions of the DOS allow the cursor and keypad keys to be configured for use with VEDIT and many other programs. We recommend upgrading to the latest version of the DOS, since it contains several other enhancements over 3.0. In the mean time, you may wish to install this patch to your DOS version 3.0, for which you may need to reference the manual concerning the use of DDT and the procedure for performing a "SYSGEN".

The keyboard produces unique codes for the cursor and keypad keys which have the high order bit set. This is excellent for the use with VEDIT. Unfortunately, the BIOS intercepts the cursor keys (and does nothing useful with them) and strips the high order bit from the keypad keys, making them useless to VEDIT. The BIOS also intercepts CTRL-W, which modifies the screen display when it is typed. Fortunately, a very simple patch cures these problems. The keypad will still produce numerics when not used with VEDIT, since CP/M and most other programs will strip the top bit themselves. This patch should therefore not affect the operation with any other programs.

The section of the BIOS being patched is not supplied in the partial BIOS listing from Intertec. In a 64K machine the code starts at E50B. (Try the command "LE50B" with DDT). The code is:

```
E50B  IN    50
E50D  IN    50        ;Don't ask us why this is repeated 3 times.
E50F  IN    50        ;Get keyboard character in A.
E511  MOV   B,A       ;Save in B.
E512  NOP             ;A very convenient patch space.
E513  NOP
E514  NOP
E515  NOP
E516  NOP
E517  CPI   17        ;Is char a CTRL-W ?
E519  JZ    E78E      ;Yes, go modify the screen display.
```

We want to change it to:

```
E50B  IN    50
E50D  IN    50
E50F  IN    50        ;Get keyboard character in A.
E511  MOV   B,A       ;Save in B.
E512  CPI   80H       ;Does character have top bit set?
E514  RNC             ;Yes, return with the character now!
E515  NOP
E516  NOP
E517  CPI   FF        ;Look for a character which does not exist.
E519  JZ    E78E      ;It will never branch away now.
```

You could apply this patch each time you use VEDIT, by creating a SUBMIT file which invoked DDT and made the patch. We leave that up to the reader. A better way is to make a permanent patch to the operating system. This is done by patching the 64CPM5/5.COM or 32CPM5/5.COM files. These files actually are SYSGEN image files. Therefore the code at E50B hex will be at 268B hex in each of these files. The complete patch is as follows:

```
DDT22 64CPM5/5.COM  or  DDT22 32CPM5/5.COM
DDT VERS 2.2
NEXT  PC
3100  0100
-S2692                    Note:  The underlined text is
2692  00   FE                     typed by the user.
2693  00   80
2694  00   D0
2695  00   00
2696  00   00
2697  FE   FE
2698  17   FF
2699  CA   CTRL-C
B>SAVE 48 64CPM5/5.COM or SAVE 48 32CPM5/5.COM
```

Now follow the instructions in your manual for performing a SYSGEN or installing the new 64CPM5/5.COM or 32CPM5/5.COM file. In brief, give the CP/M command:

64CPM5/5 <return> or 32CPM5/5 for a 32K machine

To the question "SOURCE DRIVE ..." type a <return>

To the question "DESTINATION DRIVE ..." type a "A" or "B" to save the new patched CP/M.

Now put the disk with the new CP/M into drive "A" and perform a Cold Boot. You can now customize VEDIT to use your the cursor and keypad keys. Obviously the cursor keys should be used for the cursor functions. The numeric keys can be used for any other cursor movement and editing functions.

VEDIT NOTES FOR APPLE II

The version of VEDIT supplied for the APPLE is the regular CRT version of VEDIT. It will work with the standard APPLE display as well as with most of the 24 X 80 video cards available for the APPLE. It will also work with most CRT terminals which might be connected via a serial port with the APPLE.

Installation of VEDIT is done in two steps. The first is to use the Microsoft supplied "CONFIGIO.BAS" program to configure the "Software" terminal table to either the SOROC IQ120 or the HAZELTINE 1500. If you have never changed your CP/M with "CONFIGIO.BAS", you should be able to use VEDIT by selecting the SOROC 120/140 in VEDIT's CRT menu. (There is then no need to use CONFIGIO.BAS.) The second step is to perform the VEDIT customization process, described in the manual, to configure VEDIT to the selected "Software" terminal and keyboard layout. Note that these two steps need only to be performed once. However, the customization can be performed as often as desired to produce new configurations of VEDIT.

To enable you to use VEDIT as quickly as possible, we have included a preconfigured, ready to run VEDIT in the file "VEDIT.COM". It supports the standard 40 column Apple II screen and follows the customization described in the attached sheets "Example Keyboard Layout for Apple II" and "Example Customization for Apple II". If you have an 80 column board, you can quickly customize VEDIT by performing only Task 5. of the customization.

When customizing your own keyboard layout, please note that some control characters are redefined as displayable characters. (See Redefinition of Keyboard Characters in the Microsoft manuals). By default, CTRL-K, CTRL-@ and CTRL-B are redefined as "[", RUBOUT and BACKSLASH. CTRL-K and CTRL-B may therefore not be used for Visual Functions, although CTRL-@ may be used.

Also note that the "<--" key generates a <CTRL-H>, "-->" generates a <CTRL-I> and typing <CRTL-U> actually generates a <CTRL-I>.

If an external terminal is to be used, the "CONFIGIO.BAS" program must be run and both hardware and software configuration tables configured identically, i.e. the Microsoft BIOS must do no translations. Configuring both to the default DATAMEDIA is sufficient, regardless of what terminal is connected.

IMPORTANT NOTE:

> The command mode iteration brackets, normally "[" and "]", have been customized to be "<" and ">" for the Apple II. Therefore, for all command mode examples in the manual using "[" and "]", you should type "<" and ">". If you prefer the square brackets you can select them in Task 3. of the customization.

EXAMPLE CUSTOMIZATION FOR APPLE II, CRT VERSION

A typical Apple II customization session is listed below. For clarity sake, each reply below is preceded by "--", which does not appear is the actual customization.

CRT MENU - Normally choose SOROC 120/140, unless you have used the MicroSoft supplied CONFIGIO program to select a different "Software" terminal.

```
3.)  HEX CODE FOR SCREEN CONTINUATION CHARACTER   (2D) -- 2D
     HEX CODE FOR COMMAND ESCAPE CHARACTER        (1B) -- 1B
     HEX CODE FOR COMMAND ITERATION LEFT BRACKET  (5B) -- 3C
     HEX CODE FOR COMMAND ITERATION RIGHT BRACKET (5D) -- 3E
     HEX CODE FOR SEARCH PATTERN MATCH CHARACTER  (7C) -- 7C

4.)  EXPAND TAB WITH SPACES          (0=NO, 1=YES) -- 0
     AUTO BUFFERING IN VISUAL MODE (0=NO, 1=FORWARD, 2=BACKWARD) -- 1
     BEGIN IN VISUAL MODE            (0=NO, 1=YES) -- 1
     POINT PAST TEXT REG. INSERT     (0=NO, 1=YES) -- 1
     IGNORE UPPER/LOWER CASE DISTINCTION IN SEARCH (0=NO, 1=YES) -- 1
     CLEAR SCREEN ON VISUAL EXIT     (0=NO, 1=YES) -- 0
     REVERSE UPPER AND LOWER CASE    (0=NO, 1=YES) -- 0
     IGNORE SEARCH ERRORS            (0=NO, 1=YES) -- 0
     EXPLICIT STRING TERMINATORS     (0=NO, 1=YES) -- 0

     INDENT INCREMENT  (1 - 20, SUGGEST 4)                -- 4
     LOWER CASE CONVERT (0=NO, 1=YES, 2=CONDITIONAL)      -- 0
     DECIMAL CODE FOR CONDITIONAL CONVERSION CHAR. (59)   -- 59
     LINE AND COLUMN DISPLAY (0=NONE, 1=LINE, 2=COLUMN, 3=BOTH) -- 3
     RIGHT MARGIN FOR WORD WRAP IN DECIMAL (0=OFF)        -- 0
     HIGH BIT ALLOWED ON INPUT/OUTPUT (1 - 7, SUGGEST 1)  -- 1
     CURSOR POSITIONING OPTION (0 - 2, SUGGEST 1)         -- 1
     VIRTUAL LINE LENGTH WITH SCROLLING (40 - 254)        -- 200
     HORIZONTAL SCROLL INCREMENT (1 - 100)                -- 20

5.)  ENTER NUMBER OF SCREEN LINES IN DECIMAL    -- 24
     ENTER LINE MOVEMENT FOR PAGING IN DECIMAL -- 20
     ENTER TOP LINE FOR CURSOR IN DECIMAL      -- 3
     ENTER BOTTOM LINE FOR CURSOR IN DECIMAL   -- 20
     ENTER LENGTH OF DISPLAYED LINE IN DECIMAL -- 40

6.)  SIZE IN DECIMAL OF SPARE MEMORY FOR AUTO READ     -- 4096
     SIZE IN DECIMAL OF FILE MOVE TRANSFERS IN K BYTES -- 8

     DO YOU WISH TO USE THE DEFAULT TAB POSITIONS? (Y OR N) -- Y
     ENTER DECIMAL VALUE  (4MHZ = 76, 2MHZ = 38) -- 38

     BEGIN IN INSERT MODE (0=NO, 1=YES) -- 0

     USE DEFAULT DRIVE FOR HELP AND VEDIT.INI FILES (0=NO, 1=YES) -- 0
     ENTER (0=NO) OR DRIVE # FOR HELP AND VEDIT.INI FILES         -- 1
     SHOULD VEDIT.INI FILE BE EXECUTED (0=NO, 1=YES)              -- 1

     REVERSE VIDEO ON STATUS LINE  (0=NO, 1=YES) -- 0
```

EXAMPLE KEYBOARD LAYOUT FOR APPLE II

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)

[HOME]	ESC - H	
[ZEND]	ESC - Z	
[CURSOR UP]	[CTRL-E]	
[CURSOR DOWN]	[CTRL-C]	
[CURSOR RIGHT]	[CTRL-D]	
[CURSOR LEFT]	[CTRL-S]	
[BACK TAB]	[CTRL-A]	
[TAB CURSOR]	[CTRL-F]	Useful for fast cursor movement.
[ZIP]	[CTRL-G]	
[NEXT LINE]	NOT USED	Not needed in tight layouts
[LINE TOGGLE]	NOT USED	Or use in place of [ZIP]
[SCROLL UP]	[CTRL-Q]	
[SCROLL DOWN]	[CTRL-Z]	
[SCROLL RIGHT]	[CTRL-Y]	
[SCROLL LEFT]	[CTRL-T]	
[PREVIOUS WORD]	ESC - 5	
[NEXT WORD]	ESC - 6	
[PREVIOUS PARAGRAPH]	ESC - W	
[NEXT PARAGRAPH]	ESC - X	
[PAGE UP]	[CTRL-W]	
[PAGE DOWN]	[CTRL-X]	
[SCREEN TOGGLE]	ESC - T	
[BACKSPACE]	[CTRL-H]	Or use "<---" key.
[DELETE]	[CTRL-N]	
[ERASE TO END OF LINE]	[CTRL-L]	Also called [EREOL] in manual.
[ERASE LINE]	ESC - L	
[DEL PREVIOUS WORD]	ESC - 7	
[DEL NEXT WORD]	ESC - 8	
[UNDO]	[CTRL-J]	
[TAB CHARACTER]	[CTRL-I]	Or use "--->" key.
[NEXT CHAR LITERAL]	ESC - Q	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CTRL-V]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	ESC - 1	
[REPLACE]	ESC - 2	
[CANCEL]	ESC - 3	
[COPY TO TEXT REGISTER]	ESC - C	
[MOVE TO TEXT REGISTER]	ESC - M	
[INSERT TEXT REGISTER]	ESC - I	
[PRINT TEXT]	ESC - P	
[SET TEXT MARKER]	ESC - S	
[GOTO TEXT MARKER]	ESC - G	
[FORMAT PARAGRAPH]	ESC - F	
[VISUAL ESCAPE]	[CTRL-SHIFT-P]	
[VISUAL EXIT]	ESC - ESC	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE KEYBOARD LAYOUT FOR THE XEROX 820

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)

[HOME]	ESC - CTRL-H	
[ZEND]	ESC - CTRL-Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[CTRL-S]	
[TAB CURSOR]	[CTRL-F]	Useful for fast cursor movement.
[ZIP]	[CTRL-G]	
[NEXT LINE]	LINE-FEED	
[LINE TOGGLE]	NOT USED	Or use in place of [ZIP]
[SCROLL UP]	ESC - U	
[SCROLL DOWN]	ESC - D	
[SCROLL RIGHT]	[CTRL-Y]	
[SCROLL LEFT]	[CTRL-T]	
[PREVIOUS WORD]	[CTRL-K]	
[NEXT WORD]	[CTRL-L]	
[PREVIOUS PARAGRAPH]	ESC - Q	
[NEXT PARAGRAPH]	ESC - Z	
[PAGE UP]	[CTRL-Q]	
[PAGE DOWN]	[CTRL-Z]	
[SCREEN TOGGLE]	[CTRL-W]	
[BACKSPACE]	BACK-SPACE	
[DELETE]	DEL	
[ERASE TO END OF LINE]	[CTRL-N]	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-X]	
[DEL PREVIOUS WORD]	ESC - K	
[DEL NEXT WORD]	ESC - L	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	TAB	
[NEXT CHAR LITERAL]	ESC - N	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CTRL-V]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	ESC - F	
[REPLACE]	ESC - R	
[CANCEL]	ESC - O	
[COPY TO TEXT REGISTER]	ESC - C	
[MOVE TO TEXT REGISTER]	ESC - M	
[INSERT TEXT REGISTER]	ESC - I	
[PRINT TEXT]	ESC - P	
[SET TEXT MARKER]	ESC - S	
[GOTO TEXT MARKER]	ESC - G	
[FORMAT PARAGRAPH]	ESC - B	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	[CTRL-E]	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

EXAMPLE KEYBOARD LAYOUT FOR THE OSBORNE I COMPUTER

"ESCAPE MODE CHARACTER #1"	[ESC]	
"ESCAPE MODE CHARACTER #2"	NOT USED	Type [RETURN]
"COMMON 2ND CHARACTER #1..."	NOT USED	Type [RETURN]
"UPPER/LOWER CASE ESCAPE..."	1	(1 = YES, 0 = NO)

[HOME]	ESC - H	
[ZEND]	ESC - Z	
[CURSOR UP]	[Up Arrow]	
[CURSOR DOWN]	[Down Arrow]	
[CURSOR RIGHT]	[Right Arrow]	
[CURSOR LEFT]	[Left Arrow]	
[BACK TAB]	[CTRL-A]	
[TAB CURSOR]	[CTRL-F]	Useful for fast cursor movement.
[ZIP]	[CTRL-G]	
[NEXT LINE]	[CTRL-Z]	
[LINE TOGGLE]	NOT USED	Or use in place of [ZIP]
[SCROLL UP]	ESC - U	
[SCROLL DOWN]	ESC - D	
[SCROLL RIGHT]	[CTRL-Y]	
[SCROLL LEFT]	[CTRL-T]	
[PREVIOUS WORD]	[CTRL-S]	
[NEXT WORD]	[CTRL-D]	
[PREVIOUS PARAGRAPH]	ESC - W	
[NEXT PARAGRAPH]	ESC - X	
[PAGE UP]	[CTRL-W]	
[PAGE DOWN]	[CTRL-X]	
[SCREEN TOGGLE]	[CTRL-Q]	
[BACKSPACE]	[CTRL-B]	
[DELETE]	DEL	
[ERASE TO END OF LINE]	ESC - N	Also called [EREOL] in manual.
[ERASE LINE]	[CTRL-N]	
[DEL PREVIOUS WORD]	ESC - S	
[DEL NEXT WORD]	ESC - D	
[UNDO]	[CTRL-U]	
[TAB CHARACTER]	[CTRL-I]	Or use TAB key.
[NEXT CHAR LITERAL]	ESC - Q	
[SET INSERT MODE]	NOT USED	Type [RETURN]
[RESET INSERT MODE]	NOT USED	Type [RETURN]
[SWITCH INSERT MODE]	[CTRL-V]	
[REPEAT]	[CTRL-R]	
[INDENT]	[CTRL-P]	
[UNDENT]	[CTRL-O]	
[FIND]	ESC - F	
[REPLACE]	ESC - R	
[CANCEL]	ESC - O	
[COPY TO TEXT REGISTER]	ESC - C	
[MOVE TO TEXT REGISTER]	ESC - M	
[INSERT TEXT REGISTER]	ESC - I	
[PRINT TEXT]	ESC - P	
[SET TEXT MARKER]	ESC - J	
[GOTO TEXT MARKER]	ESC - G	
[FORMAT PARAGRAPH]	ESC - B	
[VISUAL ESCAPE]	ESC - ESC	
[VISUAL EXIT]	ESC - E	Used to exit to command mode.
[RESTART EDITOR]	NOT USED	Use "EA" Command.

Note: Select ADM-3A from the menu during customization.

This page reserved for your notes.

A

A, 49, 56, 84, 90, 114, 138
Abort function, 66, 67, 71, 72
Adding new text, (See Entering new text)
Alternate command mode escape, 84
Alternate control codes, 160
ANSI standard terminal, 159
Append to text buffer, (See Concatenating Files)
Append to text register, 38, 48, 54, 70, 101, 104
Assembly language, 74
Auto-buffering, 14, 55, 56, 57, 59, 77, 89, 90, 139, 162, 168, 175
Auto-read, 14, 55, 89, 114, 138, 175, 175
Auto-startup, 58
Auto-write, 55, 77

B

B, 11, 115
[BACK TAB], 26, 78, 80
[BACKSPACE], 10, 32, 80
BACKSPACE key, 9, 10
Backup of file, 14, 130
Backward disk buffering, 55, 57, 57, 77, 90, 90, 139, 162
".BAK" file, 14, 58, 91, 132, 137, 142
Bank select, 155
Basic editing concepts, 13
Bit 8 characters, 64, 75, 174
BREAK, 46, 84, 133, 145, 184

C

C, 115
[CANCEL], 42, 43, 66, 70, 82
Cancel search operation, 42, 43, 69
"CANNOT FIND" 42, 98
"CHECKSUM ERROR", 156,172
Clear screen, 162
Close files, 129, 132, 143
Column display (status), 62, 63, 136, 164
Command categories, 18
Command escape character, 160, (See ESC key)
Command line, 17, 84
Command line editing, 84, 85, 94
Command macro, 17, 55, 58, 93, 102, 103, 121, 145, 146
Command mode, 11, 13, 13, 17, 19, 44, 84
Command mode prompt, 11, 44, 84, 94
Command syntax, 17
Comments in macros, 105, 147
Common 2nd character, 158
Concatenating files, 48

Concatenating lines, 31
Conditional convert character, 74, 136, 164
Continuation character, 62, 160
Continuation line, 16, 62, 64, 73
Control characters - in text, 41, 64, 72, 101, 119, 133, 144
Control characters - typing, 15, 22, 64, 84, 85, 159, 172, 173
CONTROL key, 22, 174
Control sequences, 15, 25, 64, 159, 172
Convert LC to UC, (See Lower to upper case conversion)
[COPY TO TEXT REGISTER], 70, 81
Copying text, 15, 45, 54, 70
Correcting mistakes, 33, (See [UNDO])
CP/M user number, 58, 88
Creating new file, 23, (See Invoking VEDIT)
CRT.ASM, 157
CRT.TBL, 156
CRT and memory mapped, 155
CRT terminal versions, 155, 155, 156, 157, 158
Cursor, 15, 18, 19, 23, 62, 67, 136, 136, 161, 162, 166
Cursor blink rate, 163
Cursor character, 161
Cursor movement, 15, 25, 26, 35, 63, 65, 67, 68, 71
Cursor position, 11, 25, 40, 44, 64, 65, 71, 72, 73, 126
Cursor type, 163
Customization, 56, 64, 64, 73, 77, 84, 94, 95, 139, 141, 154, 156
Customization notes, 172
Cut and paste, 54, 70, (See Text registers)

D

D, 116
DEC VT100 terminal, 159
Default drive, 57
[DEL NEXT WORD], 32, 72, 80
[DEL PREVIOUS WORD], 32, 72, 80
DELETE key, 67, 71
[DELETE], 10, 32, 80
Deleting files, 88, 90
Deleting lines, 31, 66
Deleting text, 10, 15, 31, 32, 66, 72, 75, 116, 120
Deleting text blocks, 15, 31, 32, 38
Delimiters (text), 93, 94, 124, 139, 163
Disk buffering, (See Auto-buffering)
Disk directory, 46, 88, 132
Disk full error (recovery), 50, 57, 58, 88, 131, 134
Disk space, 57, 68, 90, 134, 146, 162
Disk write error recovery, 90
Displayable characters, 15, 64, 65, 67
Displaying files, 47, 89, 134

E

E, 116
EA, 45, 57, 67, 87, 129
EB, 24, 45, 58, 58, 88, 89, 104, 114, 127, 130, 175, 184
EC, 48, 59, 90, 91, 131, 138, 184
ED, 46, 50, 88, 91, 132
EF, 49, 50, 91, 131, 132, 142, 143
EG, 47, 133
EI, 65, 72, 75, 119, 133
EK, 50, 88, 90, 91, 134
EL, 47, 134
EO, 100, 106, 107
EP, 36, 37, 58, 63, 73, 74, 135, 161
EQ, 51, 137
ER, 24, 48, 49, 89, 114, 130, 138
ES, 40, 55, 56, 57, 58, 68, 77, 94, 94, 139, 161
ET, 58, 141, 169
EV, 141
EW, 24, 48, 49, 50, 89, 127, 127, 130, 131, 142
EX, 12, 51, 87, 88, 129, 129, 142
EY, 45, 58, 87, 88, 89, 104, 129
EZ, 51

Edit function, 15, 25, 64, 65
Edit pointer, 18, 19, 44, 99, 115, 115, 119, 122, 124, 128, 139, 162
Edit session, 14, 23, 44, 45, 91, 129, 139
Editing second file, 45, 88, 143
Emptying a text register, 31, 38, 40, 123
End of a line, 64, 75
Ending edit session, 51
Entering new text, 10, 25, 29, 30, 64, (See Inserting text)
Entering visual mode, 11, 19, 44
[EREOL], 32, 80
[ERLINE], 32, 80
Error Messages, 182-184
<ESC> key, 11, 17, 22, 44, 84, 93, 94, 97, 103
Escape sequences, 15, 22, 64, 72, 76, 158, 159, 172, 173
Exiting VEDIT, (See Ending edit session)
Exiting visual mode, 11, 19, 44
Expand tab with spaces, 72, 139, 161
Extended commands, 116

F

F, 84, 93, 94, 115, 116, 117, 120, 122, 124, 140
File editing, 88
File extension, 14
File locking, 59
File name, 13, 105
File type, 14
Files - large (long), 14, 24, 49, 56-59, 122, 168, 175

[FIND], 42, 68, 82, (See Searching)
"Forever" (or "all"), 97
[FORMAT PARAGRAPH], 37, 73, 82, 164
Formatting paragraphs, 16, 20, 35, 37, 73, 136
Forward disk buffering, 55, 56, 57, 77, 89, 90, 128, 139
Free memory, 56, 57, 58, 125, (See U command, Memory space)
"FULL" message, 40, 46, 55, 71, 77, 125, (See Memory space)
Full screen editing, 13
Function keys, 15, 25, 154, 158, 159, 172, 173

G

G, 101, 118, 123
Global operations, 56, 58
[GOTO TEXT MARKER], 28, 79

H

H, 85, 118
Heath H19 terminal, 173
High bit characters, (See Bit 8 characters)
[HOME], 28, 77, 78
Horizontal scroll increment, 137,165
Horizontal scroll margin, 16, 66, 137, 165
Horizontal scrolling, 16, 62, 66

I

I, 93, 99, 119
I/O mapped video board, 155
IBM 3101 terminal, 173
IBM Personal Computer, 75, 165, 166
Ignore search errors, 93, 139 163
Ignored characters, 64
[INDENT], 36, 37, 71, 73, 73, 81
Indent increment, 16, 71, 136, 163
Indent position, 16, 35, 71, 71, 136
Indenting text, 16, 35, 36
Initializing terminals, 101, 144, 175
Input file, 14, 56, 57, 91, 92, 129, 131, 143, 143, 175
[INSERT], 30
Insert mode, 15, 29, 41, 64, 169
Inserting control characters, 41, 75, 133
Inserting files, 47, 89, 133
Inserting lines, 29
Inserting spaces, 68, 71, 72
Inserting text, 10, 94, 105, 119
[INSERT TEXT REGISTER], 70, 81
Inserting text register, 38, 39, 70, 118, 139
Installing VEDIT, 154, (See Customization)

Intertec Intertube II, 165
Invoking VEDIT, 9, 23, 24, 45, 58
Iteration bracket, 84, 161
Iteration count, 17, 97, 99
Iteration macro, 17, 84, 97, 98, 103, 119, 125, 141

K

K, 66, 123
Keyboard Characters, 25, 65, 74, 75, 133
Keyboard layout, 15, 22, 25, 154, 154, 158, 172
Kill files, 50, 134

L

L, 59, 99
Large files, (See Files - large)
Left margin, 73
Line and column display, 62, 164
Line continuation, (See Continuation lines)
Line number (status line), 63, 136, 164
[LINE TOGGLE], 26, 78
Literal character, 93, 94
Loading files, (See Reading files)
Loading text registers, 105, 145
Logged-in disks, 131, (See Default drive, Multiple Drives)
Long lines, 16, 62, 64, 66
Lower and upper case letters, 74
Lower to upper case conversion, 16, 74, 136, 164

M

M, 102, 103, 104
"MACRO ERROR", 94, 104, 121, 123
Macros, (See Command macros)
Mainframe computers, 68, 95
Margins, 16, 35, 37, 73, 164
MATROX video board, 166
Maximum file size, 57, (See Files - large)
Memory mapped versions, 156, 158, 175
Memory size, 168
Memory space - full, 71, 77, 85, 89, 90, 114, 118, 123, 133, 145
Memory space (saving), 40, 46, 71, 85, 90, 92, 114, 118, 127
Merging files, 47, 48, 101
Minimum transfer bytes, 56, 77, 168
[MOVE TO TEXT REGISTER], 32, 39, 40, 70, 81
Moving text, 15, 38, 39, 54, 67, 70
Multi-tasking operating systems, 59
Multiple commands, 17
Multiple drives, 24, 48, 57, 91

Multiple files, 55, 91
Multiple screen lines, 62, 66, (See Continuation lines)

N

N, 84, 90, 93, 128, 140, 175
NEC APC, 75
"Nearly" full, 56, 90
"NEW FILE", 9, 23, 130, 184
NEWCRT.DOC, 157
[NEXT CHAR LITERAL], 41, 65, 72, 75, 80, 94
[NEXT LINE], 26, 78
[NEXT PARAGRAPH], 27, 72, 79
[NEXT WORD], 27, 72, 79
"NO DIR SPACE", 50, 88, 90
"NO DISK SPACE", 50, 88, 90, 143
"NO OUTPUT FILE", 143
Normal mode, 15, 29, 64, 169
Notation, 84
Numeric pad, 173, 174

O

Offset paragraph, 73
Opening a file, 14, 130, 138, 142
Output file, 14, 55, 56, 57, 57, 77, 90, 91, 92, 114, 127, 129, 130, 131, 142, 143, 143, 175
Output to printer, (See Printing text)
Over-writing text, 10, 15, 25, 29, 30, 64, (See Normal mode)

P

P, 101, 118, 120
[PAGE DOWN], 28, 65, 79, 166
[PAGE UP], 28, 79, 166
Page movement, 28, 34, 166
Paragraphs, 68, (See Format Paragraph)
Parameters - setting, 58, 135, 139, 160, 168
Pattern Matching, 93, 95
[PREVIOUS PARAGRAPH], 27, 72, 79
[PREVIOUS WORD], 27, 72, 79
Print formatter, 20, 73, 74, 107, 141
Print keyboard layout, 170
PRINT.EXC, 107
[PRINT TEXT], 41, 72, 82
Printing text, 11, 41, 72, 100, 101, 106, 135
Printing text registers, 101
Processor speed, 169
Programmable function keys, 58, 101, 144

Q

Quick customization, 158
Quit edit session, 51, 137

R

R*, 105, 147
RD, 101, 144
RI, 93, 105, 144
RL, 101, 104, 145
RP, 101, 145
RS, 101, 104, 146
RT, 101, 104, 144, 146
RU, 101, 147
Re-routing console output, 106
READCRT.DOC file, 157
Reading files, 14, 55, 89, 131, 138
Reformat paragraph, (See Format paragraph)
Register, (See Text register)
[REPEAT], 34, 42, 43, 65, 69, 81, 82
Repeat function, 34, 65
Repeat value, 34, 65
Repeated commands, 97, (See Iteration macro)
[REPLACE], 43, 69, 82
Replace text, (See Search and Replace)
[RESET INSERT MODE], 80, 159
[RESTART], 82, 87
Restart edit session, 45, 58, 129
RETURN key, 15, 17, 29, 35, 64, 71, 75, 84
"REV FILE OPEN", 92
Reverse upper & lower case, 140, 162
Reverse video, 62, 170
Right margin, 73, 164
RUBOUT (DELETE) key, 85, 157, 160

S

S, 93, 98, 99, 124, 140
Sample edit session, 9
Save text on disk, 11, 14, 45, 101, 129, 143, 146
Save text register, 101,103,146
Scratch pad buffers, (See Text registers)
Screen address, 155, 166
Screen line length, 73, 166
Screen parameters, 165
Screen size parameters, 174
[SCREEN TOGGLE], 27, 79
Screen updating, 62, 65
[SCROLL DOWN], 27, 78

[SCROLL LEFT], 27, 79
[SCROLL RIGHT], 27, 79
[SCROLL UP], 27, 78
Scrolling of screen, 16, 29
Search (ing), 16, 19, 42, 68, 85, 140
Search and Replace, 16, 19, 43, 68, 98, 99, 103
Search errors, 93, 94, 140
Search for <ESC>, 94
Search options, 42, 69, 93
Selective replace, 43, 69
Separator (pattern), 95
".SET" file, 155, 156
[SET INSERT MODE], 10, 80, 159
[SET TEXT MARKER], 28, 79
Signon message, 170
Spare memory for auto-read, 168, 175
Splitting files, 49
Splitting lines, 11, 15, 29, 31, 64
Status line, 9, 23, 38, 62, 70, 136, 161, 164, 165, 170
Strings (of text), (See Text strings)
Strip comments, 96
Structured languages, 71
Substitute, (See Search and replace)
Suppressing <CR>, 105, 149
Suppress search errors, 93, 139, 163
[SWITCH INSERT MODE], 10, 80, 159
Switches - setting, (See Parameters - setting)

T

T, 84, 99, 100
[TAB CHARACTER], (See Tab character)
Tab character (key), 16, 35, 64, 67, 84, 119, 139
[TAB CURSOR], 26, 68, 78, 159
Tab expansion, 68, 71, 139, 162
Tab positions, 16, 19, 58, 64, 67, 68, 71, 141, 162, 169
Televideo terminals, 158, 159, 165
Temporary disk file, 57, (See ".R" file)
"TEXT" message, 63, 70
Text buffer, 14, 55, 56, 56, 57, 70, 77, 114, 125, 143
Text markers, 15, 28, 65, 67
Text registers, 15, 19, 38, 45, 46, 54, 70, 72, 101, 118, 121, 123
125
Text registers (emptying), (See Emptying text register)
Text registers (insertion), (See Inserting text register)
Text registers (loading), (See Loading text registers)
Text registers (saving), (See Saving text registers)
Text register macros, (See Command macros)
Text strings, 84, 93
TRS-80 Model I, 74
TRS-80 Model II, 166
Typing out text, 125, 146, (See Displaying files)

U

U, 85, 101, 114, 125, 147, 175
[UNDENT], 36, 37, 71, 73, 73, 81
[UNDO], 31, 33, 65, 80
Upper and lower case, 16, 117, 139, 140, 159, 173
Upper/lower case search, 117, 136, 162
User numbers (CP/M), (See CP/M user numbers)

V

V, 11, 44, 126
V-PRINT Print Formatter, 20
V-SPELL Spelling Corrector, 20
VDSETCRT.COM file, 156, 171
VEDSET.COM file, 156
VEDIT.INI file, 58, (See Auto-startup)
Version number, 141
VHELP.TXT file, 85
Video board initialization, 167
[VISUAL ESCAPE], 11, 44, 44, 51, 82, 98, 126
[VISUAL EXIT], 44, 44, 82, 98, 126
Visual functions, 15, 31, 64, 65
Visual mode, 9, 15, 19, 23, 62, 77, 98, 126, 139, 140, 162

W

W, 46, 49, 56
Wildcard characters, 88, 95, 132, (See Pattern matching)
Window into text, 56
Word processing, 20, 71, 72, 73, 141
Word wrap, 15, 35, 37, 73
Word wrap column, 136, 164
Write errors, (See Disk full errors)
Writing files, 70, 143, (See Saving text on disk)

X

XA, 106
XS, 106
XT, 106

Y

YL, 105, 106
YR, 105
YT, 93, 105
YW, 105

Z

Z, 59, 128
[ZEND], 28, 78
Zenith Z19, Z100 terminals, 173
[ZIP], 26, 78

".$$$" file, 132, 137
".R" file, 57, 88, 90, 90, 91, 182
"1 END" message, 70

< and >, 17, 84, 161
[and], 17, 84, 97, 161
@, 93, 117, 119, 124, 140
#, 17, 97, 117, 124
$, 84
/, 93
¦, 95

<CR> and <LF>, 64, 84, 95, 115, (See <CTRL-N>)
<CTRL-C>, 135, 157
<CTRL-H>, 85, 94
<CTRL-I>, 67
<CTRL-N>, 69
<CTRL-Q>, 85, 94, 119
<CTRL-R>, 85, 94
<CTRL-S>, 47, 86
<CTRL-U>, 85, 94, 157, 160
<CTRL-X>, 85
<CTRL-Z>, 65, 75

CompuView

CompuView Products, Inc.
1955 Pauline
Ann Arbor, Michigan 48103
(800) 327-5895 (313) 996-1299
Telex: 701821