Ved

Text Editor

Bob van der Poel Software

P.O. Box 57	P.O. Box 355
Wyndell, B.C.	Porthill, ID
Canada VOB 2N0	USA 86853-0355

```
***************************************
*                                     *
*                                     *
*       VED, OS-9 Visual Editor       *
*     OS-9 Level 2/CoCo 3 Version     *
*              Version 2              *
*                                     *
*       Copyright 1988 and 1989       *
*      Bob van der Poel Software      *
*                                     *
*                                     *
***************************************
```

<u>* Important **</u>

Please backup your original diskette and put it in a safe place <u>before you continue</u>. Since this diskette is not completely formatted please use the OS-9 script file “makecopy” included on the master diskette to transfer all the files from the master disk to a freshly formatted one.

This Software is offered for sale on an “as is” basis. No guarantees are made or implied.

Introduction

Thank you for purchasing this software. We hope you will find it useful. Please show the program to your friends. If they like it, ask them to purchase their own copy. Continued customer support will enable the author to develop more programs for you. Continued sales also help to feed the family. Please do your part to stop software piracy…it hurts everyone—especially you, the biggest loser when no more software is developed for your computer!

Unlike much other software you purchase today this package is not copy protected. In fact we encourage you to make as many copies as you need for your own personal use. But please be fair with us too. Don’t give copies of this program away to your friends—after all, if it’s good enough for you to spend your money on, why shouldn’t they do the same?

Getting Started

If you examine your distribution disk you will find several files in the CMDS directory, *Ved*, is the editor. It should be copied to the CMDS directory on your main system disk. IMPORTANT: Do not merge *Ved* with other program modules, if you do you will decrease the buffer memory available for *Ved*. The file, *Ved..Help*, is a stand alone program, accessible from *Ved*, containing the help files. This program should also be placed in the CMDS directory of your main system disk, or you can load it into memory with other utilities. Note: merging *Ved..Help* with *Ved* is possible (and perhaps even reasonable for novice users) but it will reduce the buffer size available by 8k.

To edit a file, simply type:

```
   Ved filename <enter>
```

where *filename* is the name of the file you wish to edit. Examples of valid filenames include:

```
   myfile
   /d0/myfile
   src/letter
```

In the normal OS-9 manner the first example edits a file in the current data directory, the second a file in the root directory on drive 0, and the third a file in a subdirectory in the current data directory.

As an option, you can specify two filenames on a command line. The first filename will be considered to be the *Infile* and the second the *Outfile.* In this case the text is initially read from *Infile*. Later, when the text is saved, it will be written to *Outfile*. An example of this usage would be:

```
   Ved filename tempfile <enter>
```

Whenever you use *Ved* you must use a filename. If you do not *Ved* will generate an error message. If the file does not exist then you will be asked if it is okay to use that filename. This prevents having a bunch of empty files created when you type in an incorrect name or think you are in a different directory than you really are. If the file does not exist, it will not be created at this point—files are not created until you exit *Ved* (see *ALT-Q* below).

If the file is found it will be read into *Ved's* buffer. This buffer automatically expands as your file grows. You do not need to use memory modifiers. The maximum size file *Ved* can handle is approximately 53,000 characters. If the file you attempt to edit is larger than this size an error message will be generated and *Ved* will terminate.

After loading a file *Ved* scans it for illegal control characters. These characters, if found, will be removed from the file. This includes line-feed and tab characters as well as graphics characters, etc.

Whenever *Ved* is started up, the current data and execution directories are checked for a file called *Ved..Defs*. By checking the data directory first, it is possible for you to have many different *Ved..Defs* files—a generic one in the execution directory and a special purpose one for a project in a data directory. If found this file will be used to initialize *Ved's* default modes, tab settings and macro definitions. If *Ved..Defs* is not found, then *Ved's* internal defaults will be used. You can create your own *Ved..Defs* files with custom information. See ALT-/,*MakeVdefs*,*VDefsText* below.

Note: For a *Ved..Defs* file to be found in the execution directory it must have the "E" attribute set. See the ATTR command in your OS-9 reference manual.

Ved will work only in true CoCo3 window screens. It will not work in VDG screens (the standard 32x16 text screens are "window"). The smallest window which *Ved* will function in is 8 lines by 30 columns—smaller sizes will generate an error.

A note on how *Ved* handles files: First, *Ved* only works on files which fit into its internal memory buffer. If the file you are attempting to edit is too large to handle in memory you will be unable to load it with *Ved*. For the case we have supplied a simple "splitter" to break up large files (see *VSplit* below for details).

Second, when the files are saved they will overwrite the existing files. No temporary files are created. The advantage to this that you can use *Ved* on a nearly full disk; attributes and filename capitalization are preserved; and it is much faster to overwrite a file than it is to create a new one and delete an old one. The disadvantage is that you original file is lost—if this is a concern, use the optional *Outfile* parameter with *Ved*.

The following sections deal with *Ved's* many commands. In order to get the most use out of *Ved* we suggest you read each section carefully and experiment with the various commands with your own files.

Ved commands are made in three ways. Some commands require only a single keypress. These are shown in the format "ENTER", "UP ARROW" etc. Most commands require holding down two keys at the same time. These are shown in the format "ALT-Q", "CTRL-A", "SHIFT-UP" etc. This means that in order to invoke the command *ALT-Q* you should press the <ALT> key with one finger and, at the same time, press the <Q> key. A third set of commands require a three key sequence. These commands always require you to first press both the "CTRL" key and the "O" key at the same time. *Ved* will then wait for a "command" key to be pressed. For example, the command *CTRL-O-LEFT* (which moves the cursor to the start of a work) is invoked by first pressing the “CTRL” key and the “O” key at the same time, then pressing the LEFT ARROW key.

Communication between you and *Ved* is done through dialogue windows which pop up in the center of the screen. If you are using a 128K CoCo and have other processes running you may not have enough memory for these windows. *Ved* will then generate a 207 error. If this occurs you should start over with less processes running.

Whenever you enter text in reply to a prompt you can immediately cancel the current operation (and return to the editor) by pressing CTRL-BREAK. When entering a line for “Find,” etc, you can correct the last character typed by pressing the LEFT ARROW key and then making the correction. Press ENTER when you are finished typing your response.

Editing Commands

When *Ved* is started it is normally in INSERT mode. This means that any characters typed are inserted into the buffer at the current cursor position. The cursor is the solid, inverse video block. Note that the squiggle character at the top of the screen (if you are editing new file) indicates the end of the file.

To add characters to a file simply move the cursor to where you want to type and TYPE! Each character will be inserted into the buffer and, of course, the results are displayed on the screen.

If you are inserting characters into an existing line, you may notice that the end of the line drops off the edge of the screen. Don’t be alarmed. Your text is still there. It will be redisplayed when any of the following occurs:

1. the cursor moves to the end of the screen line
2. the internal line buffer (about 200 characters) fills up
3. you execute a command (find, delete word, etc.)
4. you move the cursor to a new line.

This may all sound a bit confusing, but you’ll get used to it in no time. Remember, the only time text disappears is when you are inserting into an existing line.

ENTER	Inserts a carriage return into the buffer Carriage returns are displayed as a small “o” on the screen.
BREAK	The BREAK key is used to delete the character at the current cursor position.
SHIFT-BREAK	Delete the character before the cursor—a backspace+delete.

ALT-SPACE	Insert a space at the current cursor position without moving the cursor. (This is very handy when in overstrike mode.)
ALT-Y	Delete (yank) the word at the current cursor position. This command will always delete at least one character, deleting continues until the end of the current word. A word is defined as a series of characters ending in a *Block Marker*, a *SPACE*, or a terminating punctuation (.,;:?!). If you are in the middle of a word, the current word will be deleted till its end. Unlike *ALT-L* (below) *ALT-Y* will not delete the *SPACE* following the word.
ALT-L	Lift word. This command is very similar to *ALT-Y*, however it first moves the cursor to the start of the current word (a character following a *Carriage Return*, *SPACE*, or *Block Marker*. *ALT-L* will also delete the *SPACE* following the word.
ALT-K	Delete until the right edge of the screen (the current screen line).
ALT-V	Delete until the next carriage return. Since this could entail deleting several screen lines you are first asked if you are sure you want to do the deletion. Pressing any key other than <Y> cancels the operation. Note that if the current screen line ends in a carriage return then *ALT-V* is identical in operation to *ALT-K*, no “sure?” is displayed.
ALT-W	Toggles wordwrap. When *Ved* starts wordwrap is turned on. This means that any words which will not fit on a screen lone will be “wrapped” to the next screen line. Note *Ved* DOES NOT insert carriage returns into the text in order to do this—the only time a carriage return is entered into the text buffer is when you press the enter key. As you edit your text the points at which wordwrap occurs changes automatically, you never need to concern yourself with it. In most cases you should leave *WordWrap* on—leaving it off will force extra screen updates when block markers are deleted.
ALT-I	Toggle auto-indent mode. This feature will be appreciated by people using *Ved* to write programs in a structured language (like Pascal or C) which uses various levels of indentation. Of course, it can also be used to create tables, etc. When auto-indent is on, the cursor will be advanced to the same position on the new line as the first non-space

character on the previous line.

ALT-O Toggles overstrike/insert modes. When *Ved* starts up it is in insert mode. *ALT-O* will change to overstrike mode. When in overstrike the cursor will be changed from the normal block to an underline. When in overstrike the character at the cursor will be changed to the key you press, in insert the key pressed will be inserted in front of the character at the cursor. If the cursor is at the end of file or at a carriage return, characters will always be inserted, just as if you were in insert mode.

Cursor Control

UP ARROW Moves cursor up one line.

DOWN ARROW Moves cursor down one line.

LEFT ARROW Moves cursor left one character.

RIGHT ARROW Moves cursor right one character.

SHIFT-UP ARROW Moves cursor to top of screen. If the cursor is currently at the top of the screen it will be moved to the bottom of the previous screen.

SHIFT-DOWN ARROW Moves cursor to bottom of the screen. If the cursor is currently at the bottom of the screen it will be moved to the top of the next screen.

SHIFT-LEFT ARROW Moves cursor to start of the previous word.

SHIFT-RIGHT ARROW Moves cursor to start of the next word.

CTRL-O-LEFT ARROW Moves the cursor to the start of the current word. If the cursor is on a *SPACE*, *Carriage Return* or *Block Marker* no action will be taken. This command is useful for macros which attempt to mark a word with word processing marks—see the file *VedDefs.txt* for examples.

CTRL-O-RIGHT ARROW Moves the cursor to the end of the current word. If the cursor is on a *SPACE*, *Carriage Return* or *Block Marker* no action will be taken. The end of a word is defined as a *SPACE*, *Carriage Return* or a terminating punctuation (.,;:!?). Again see *VedDefs.txt* for examples.

CTRL-UP ARROW Moves cursor up one screen. Essentially this is a paging command…much quicker than using *UP ARROW* to scroll through the file. Note that the top line of the current screen is redisplayed at the bottom of the new screen and that the cursor stays on the same screen line.

CTRL-DOWN ARROW Moves the cursor down one screen. Note that the bottom line of the current screen is redisplayed at the top of the new screen.

CTRL-LEFT ARROW Moves the cursor to the start of the current screen line.

CTRL-RIGHT ARROW Moves the cursor to the end of the current screen line.

ALT-UP ARROW Moves cursor to the start of the file.

ALT-DOWN ARROW Moves cursor to the end of the file.

ALT-LEFT ARROW Moves the cursor to the start of the current line. Note that the start of the current line is not necessarily on the same screen line. Lines are defined as a string of text starting and ending with a carriage return and can span several screen lines.

ALT-RIGHT ARROW Moves the cursor to the end of the current line. (See note above.)

Global Commands

When entering text for the following commands the following keys serve special functions:

ALT-ENTER When included in a search or replace pattern this generates a carriage return character.

ALT-- (Alt-minus) This key generates a ± character. This is a wild-card character which will match any character. For example “mo±e” will match both “mole” and “more”.

Following are the global commands:

ALT-F Find a pattern. First you will be asked to enter a pattern. The buffer will be scanned for a match staring at the cursor position. If a match is found the cursor will be moved to the start of the match and the match will be highlighted. Pressing a key will terminate the highlight. No keypresses will be lost. If no match is found a “no match found” message will be displayed.

ALT-N Finds a pattern previously entered for ALT-F. Very useful in those cases when the initial *ALT-F* finds the wrong match.

ALT-R Find a pattern to replace with new one. You will first be asked for a pattern to find (“original pattern”) and the pattern to replace it with. Next you will be asked if you wish to replace only the next occurrence of the pattern, all occurrences or prompted. If you press <N> only the next occurrence of the pattern will be changed. If you press <A> then all occurrences of the pattern, from the current cursor position, will be changed. If you press <P> then the cursor will be moved to the next occurrence of the pattern, the pattern will be highlighted until you press <Y>, <N> or <BREAK>. <Y> will change the pattern and advance the cursor to the next match, <N> will not change the pattern, however the cursor will be advanced to the next occurrence, <BREAK> will stop the operation of replace. If you are doing a prompted or global (all) replacement, the number of changes made will be displayed when the end of file is reached and the cursor will return to the position it was when you started the replace. If no matches at all are found then a “no match found” message will be displayed.

Replace will stop its operation when it encounters an *End Block* marker. This is useful if you only wish to make “global” changes to part of a file.

You can use a ± in both the original pattern and the replacement pattern. A find/replace of:

Original Pattern	c±t
Replacement Pattern	d±g

will make the following changes:

cat	dog
cot	dog
cut	dug

Using a wildcard in a replace string only makes sense if a wildcard is specified in the same position in the find string, however *Ved* will accept it in either or both strings in any position(s).

The most likely way wildcards in replace strings will be used is in cases where you are editing a file with carriage returns in it and you want to make changes to two-word combinations which may or may not span a line and you wish to keep the original carriage returns.

ALT-A Duplicates the last *ALT-R* entered. Note that this includes the mode (N,A or P).

ALT-P We're not sure if this command belongs here or not, but does effect all the text, so it can be called "global". *ALT-P* will "format" the buffer with carriage returns to a specified line length. This can be very useful if you are preparing messages to be uploaded to a BBS, or for a text formatter which cannot handle long lines of text. But be careful! The effects of *ALT-P* cannot be undone. This is because the carriage returns inserted by *ALT-P* are identical to the ones you generate with the <ENTER> key (See also comments under *ALT-N*.) *ALT-P* will prompt you for the line width you want. Enter a number between 32 and 250. Any other vaule will generate an illegal value error.

ALT-J Jump to label or line number. This command is very useful when using *Ved* to prepare assembler (or other programming language) source files. *Jump* prompts for a line number or label. If you enter a number *Ved* will move the cursor to that line number; note that this is actual lines, not screen lines. If you do not enter a number then *Ved* attempts to fins a line which begins with your input text. The text must match exactly and be terminated with a non-alphanumeric character. The search is done from the start of the file. Entering "start" would position the cursor to any of the following lines:

```
start
start:
start lda #1
```

If more than one of the above lines are present in your file then the cursor will advance to the first one. Note that a jump to "start" will not find the following:

```
starting lda #1
start1: lda #2
the start,
```

Since the search being doneis for text FOLLOWING a carriage return the first line in your file will never generate a match.

ALT-; Toggle case sensitivity. When *Ved* starts up it is in a “case-sense” off mode. This means that *Ved’s* searches (*Find*, *Replace* and *Jump*) will treat upper and lowercase characters as being identical. For example, a find for “cat” will find “Cat” and “CAT” as well as “cat”. Using ALT-; to toggle case-sense on forces *Ved* to treat upper and lowercase letters differently. Now a search for “cat” will ignore “CAT”. In addition, with case-sense off *Replace* will attempt to maintain the original case of replacements. For example, changing “cat” to “dog” will change “Cat” to “Dog” and “CAT” to “DOG”.

Block Commands

Ved, like any good editor, permits a number of operations on blocks of text. This includes moving, copying and deleting. In order for any block operation to function, the block must first be defined.

ALT-, Insert a *Begin-Block* marker. This is easier to remember as an ALT-< since it inserts a marker in the text which indicates the start of a block. Note that this marker can be deleted just as any other character. If a *Begin Block* marker already exists the existing one will be deleted.

ALT-. Insert an *End Block* marker. Again, remember this as ALT->, this inserts an end of block marker. If an *End Block* marker already exists the exiting one will be deleted. When the cursor is on a *Block Marker* it will disappear. Sorry, but there is no easy way (considering memory restraints and some bugs in the OS-9 graphics drivers) which can give us a “cursor” over a marker. The only solution is to use a character out of the extended character set to mark blocks, etc. This method was used by *Ved 1.x*, but then the markers were hard to find on the screen. In discussions with many users it was decided that the current method is most workable.

ALT-Z	Delete all block markers. This command is seldom needed in normal operation since existing markers are deleted when new ones are inserted. However, it can be useful in macros (see the supplied file *VedDefs.txt* for examples).
ALT-B	Jump to a *Begin-Block* marker. If no marker exists then the command is ignored.
ALT-E	Jump to an *End-Block* marker. If no marker exists then the command is ignored.
ALT-C	Copy the marked block. This command will generate an error message if both the begin and end marker are not present (and in the correct order) or if the cursor is currently in the defined block. Copy expands the buffer and places a copy of the marked text at the current cursor position. The existing block markers are not deleted, making it easy to do multiple block copies. Use *ALT-Z* to delete markers after you are finished with a block.
ALT-M	Move marked block. This command will generate an error message if both the begin and end marker are not present (and in the correct order) or if the cursor is currently in the defined block. Move takes the marked text and moves it to the current cursor position. If the marked block is less than 2000 characters then *Move* uses internal buffers, however if the block is larger than 2000 characters a temporary disk file is created in the current data directory (if you are doing a lot of large moves you might want to use *ALT-H* to set up a RAM-Disk as your current data directory). This file is deleted when *Move* is completed. Also, note that unlike *Copy*, *Move* deletes the block markers.
ALT-D	Delete marked block. This command will delete a marked block of text. Note that before deleting the block the cursor is first moved to the start of the block and you are asked if you are sure you want to delete the text. In addition to the text, the block markers are also deleted.
ALT-S	Save marked block. The marked block will be saved to the filename you specify. Note that specifying “/p” will print the marked block to the printer. *ALT-S* does not effect the block markers.

ALT-G Get a block of text (append). This is the only block command for which you do not need block markers. *ALT-G* will prompt you for a filename and append that file to the file currently in memory. Note that the append is done at the current cursor position, not the end of the file.

Macros

By permitting the user to define a frequently used series of keystrokes into a single “macro” key, *Ved* lets you get more done in your limited time. What can macro keys do for you? Well, if you are writing a story you might want to put the name of your characters into macro keys—not only is this faster, but it also ensures that they are spelled the same each time they are used; if you are preparing a complex document you might want to use a macro key to insert formatting commands in your file; finally, you can use macro keys to do complex editing.
Ved has three different kinds of macro keys; Single key user defined, two key user defined and predefined.

ALT-8 Number Macro. To aid in writing documents with points (or programs needing line numbers) *Ved* supplies a number register. Pressing *ALT-8* will invoke the number macro, inserting the current contents of the number register into the text. After each *ALT-8* the number register is incremented by 1. The number register is initialized to 0. The number register is an un-signed 16 bit value (0..65535). Includeing *ALT-8* in a macro definition can make setting up numbered points very easy.

ALT-7 Initialize number register. The number register can be set at anytime with *ALT-*7. You will be prompted for a number, this number will be evaluated and (if valid) set in the number register. After the number is evaluated you will be prompted for an increment. If you wish to change the increment value used each time an *ALT-8* is done, enter a value. If you do not enter a number the increment will be left the same as it was originally (default on starup is 1). If you want to get real tricky you can enter 65535 as the increment value and *ALT-8* will decrement by 1 each time; 65534 will decrement by 2, etc.

CTRL-O-* Filename macro. The current filename will appear and be printed. This macro can be useful when creating shell macros which use *VPRINT* to print the text in memory. See the file *VedDefs.txt* for examples.

ALT-: Define a single key macro. You will first be asked which macro key you wish to define. The keys CTRL-A, B, D, F, G, K, N, T, U, V and W are permitted macros. Even though you must use the CTRL key to use a macro, just press the key to define it. You can type in up to 28 characters in your macro definition. This includes the ENTER key, cursor keys (LEFT ARROW, etc.), *Ved* commands (*ALT-F*, *ALT-SPACE*, etc.), other macros and normal text characters. Since the LEFT-ARROW key can be included in a definition, you cannot use it to correct a mistake in entering a macro definition—if you make a mistake you’ll have to end the entry and type it in again. To end the entry of a macro press CTRL-BREAK. See the section below on *MakeVDefs* for an alternative method of creating macros.

To use a single key macro, simply press the CTRL key and the key corresponding to the macro key. The macro will be executed just as if you had typed it from the keyboard.
Note that macro keys can reference other macro keys (CTRL-A is permitted in the definition of macro “V”). Macro calls can be “nested” to a level of 9 deep. This means that CTRL-A could include a call to itself forming an infinite loop. This can be useful in creating editing macros. All macros terminate if an error is encountered.

To take complete advantage of macro keys we suggest you spend some time playing with various combinations. We’re sure you’ll like them!

ALT-- Define a two key macro. This command is essentially the same as *ALT-:*, however the macro defined requires a two key sequence to invoke it. After pressing *ALT—* you will be asked which “O-Macro” you wish to define. Enter any key from “A” to “Z” and enter the macro as described above.

A macro defined with the *ALT—*command can be invoked by pressing *CTRL-0* followed by the macro name (“A” to “Z”). This somewhat complex arrangement is needed to expand the number of macro keys available to the user and the limited number of keys on the Color Computer.

Tabbing

ALT-ENTER Move the cursor to the next tab stop. If a carriage return is encountered before enough positions have been moved over, space characters will be inserted into the text. If the cursor is currently past the last defined tab stop then *ALT-ENTER* will be ignored.

ALT-T Define/view tab stops. When this command is executed a “ruler” will appear on the screen. The “X”s on the top line indicate tab stops. Use the LEFT and RIGHT ARROWs to move the cursor to the position you wish to set/delete a tabstop. Use the SPACEBAR to toggle tabstops on and off. When you have the tabs set to your liking press BREAK or ENTER to return to the main editor.

Miscellaneous Commands

ALT-U Undo has 2 incarnations. First, if you are editing a line *ALT-U* will restore the line to what it was before changes. Note that any command will reset the current line and that if you do an *ALT-U* just after typing a new line then that line will be deleted—actually it’s restored to what it was before the edit, which is I nothing. So be careful hitting *ALT-U*.

In order to force the start of a new line buffer hit *ALT-BREAK*. This is essentially a do nothing command, however it forces the current line into the main buffer where it is saved from an accidental *ALT-U*. Note that I *ALT-BREAK* also brings back the text forced off the screen during inserting.

Second, all line and word deletes are now saved in a buffer. You can restore a deleted work or line at anytime with *ALT-U*. The data will be placed back into the buffer at the current cursor position, not where you deleted it.

Lines deleted with *ALT-V* are not saved if you get a ‘sure?’ message, however they will be saved if the operation of *ALT-V* and *ALT-K* are the same.

We find a combination of *ALT-K* and *ALT-U* very useful for moving one line of text to a new position.

ALT-0 Case switch. This command will change the case of the word under the cursor. When an *ALT-0* is done the cursor will be moved to the start of the word. Now one of three things will happen, depending on the current mix of upper and lower case letters in the word:

1. If the first letter of the word is a lowercase “a”…”z” that letter will be changed to its uppercase equivalent.

2. If the first letter of the word is an uppercase "A"…"Z" and the second character is a lowercase "a"…"z" then the entore word will be changed to uppercase characters.
3. If the first two characters of the word are uppercase "A"…"Z" then the entire will be changed to lowercase.

This sounds much more complicated than it is. Why not just put the cursor on a word and hit *ALT-0* a few times to see what happens. We're sure you'll find it very handy for fixing up those capitalization errors.

ALT-/ Save current defaults (case-sense, overstrike, wordwrap and auto-indent), tab settings and macro definitions. These items are saved to the file *Ved..Defs* in the current data or Execution directory. You will be asked if you wish to save to the Data or Execution directory. Press "D" or "E"; any other keystrokes will cancel saving of the defaults. If an existing file exists, it will be overwritten, otherwise a new file will be created.

CTRL-O-/ Reload defaults. This command will first prompt you to make sure you want to reload the defaults. The current data directory is first checked for the file *Ved..Defs*. If not found there the current execution directory is checked. No error is generated if *Ved..Defs* cannot be found. This command can be handy if you have been "fooling around" with macros and wish to restore things to a known setting.

CTRL-O-@ Reload the current file. Again, handy if you've been "fooling around," this command will reload the file specified when you started *Ved*. If you have specified an *outfile* this is the filename which will be searched for. If the file does not exist (quite possible if you haven't yet done a save and are using both *infile* and *outfile*) you will receive an OS-9 error. In this case there will be no change to your current file.

ALT-@ Display a ruler. This is very useful when creating tables, etc. The ruler will be displayed on a line directly above the current cursor position. We hope that you'll find that a rule displayed in this manner is much more useful than confusing status lines which reduce the amount of text which can be displayed at any one time.

ALT-X This command will permit you to save your current buffer, without leaving *Ved*. When you do an *ALT-X* you'll see a prompt for a filename along with the original filename. If you press <ENTER> you will save the current buffer to the current filename. If you enter a new filename the buffer will be saved to that file. If the file already exists you'll be asked if you are sure—you must press <Y> to overwrite the existing file; if you do not press <Y> you'll start the Xsave routine over.

ALT-Q Quit editing. Quit will first ask if you wish to save the file. Replying with anything other than <N> will start the following process:

1. The current buffer will be saved to a temporary file,
2. the original file will be deleted,
3. the temporary file will be renamed to the original filename,
4. *Ved* will terminate.

Replying with any character other than <Y> will present an "are you sure" prompt. You must then press <Y> to force *Ved* to terminate without saving its buffer. In this case the original file will be unchanged, and no temporary files will be created.

ALT-9 Starts an OS-9 shell. This lets you view directories, etc. or even start a completely new task.

There are two different ways this command can be used. First, if you enter a command at the OS9> prompt a sub-shell will process this command. When the command is completed you will return to *Ved*. On the other hand, if you just press <ENTER> at the OS9> prompt the sub-shell will accept commands until ESCAPE (CTRL-BREAK) is pressed. In either case a "press any key" message will appear before the screen is refreshed—this will permit you to view the contents of a directory, etc. before the editor screen reappears.

This shell gateway can be used to interface to our *VPRINT* text formatter. One simple example is the "print file" macro in the sample macro file. This macro saves the current text buffer (*ALT-X*), starts a sub-shell (*ALT-9*) and passes the name of the formatter and the current filename as the parameter (Vpt *CTRL-0-**).

ALT-H	Change current data directory (chd), Changing the current data directory will effect the paths used for *ALT-G*, *ALT-/* and *ALT-M* (if external buffers are needed), as well as the directory used for saving the text buffer after an *ALT-Q* if a complete path list is not specified. For this reason you should be very careful in using *ALT-H* unless you know exactly what you are doing.
F1	Generates a current status summary. Note that "lines" refer to a string of characters terminated by a carriage return, not screen lines.
SHIFT-F1	Generates a status window with the current default settings.
SHIFT F2	Help summaries are available at any time by pressing this key. In order to function the program *Ved..Help* must be in memory or in the current execution directory. First select the number key (1..5) corresponding to the area you need refreshing in, then after the summary press any key to return to *Ved*.
ALT-6	Insert a position marker in the text file. This marker (in reverse video "^") can be used to mark a position in the text you wish to return to later. If a position marker already exists in the file it will be deleted.
ALT-5	Jump to a position marker inserted previously with *ALT-6*. If no marker exists the command will be ignored.

Error Messages

From time to time *Ved* will report an error. *Ved* errors are broken down into two types: OS-9 system and *Ved* internal. OS-9 system errors (most likely when accessing files and devices) will be reported in standard OS-9 fashion (the error code number, or if a enhanced error driver has been installed, a text message). Internal errors are shown as "Ved Error!" and a description of the error. Most should be self-explanatory, however we will expand on "Can't expand buffer, out of memory."

This error will occur if *Ved* is unable to expand it's interal buffer to accommodate more text. This can occur because you are truly out of memory, or if *Ved*'s buffer has approached the 53,000 character limit. When this error occurs you will have trouble even deleting characters. This is due to the fact that character deletions are done in a line buffer, and there must be room to expand the text buffer to the full size of the line buffer. In this case you must use a command like *ALT-K*, *ALT-V* or *ALT-Y* to delete characters. Once a line or two have been deleted you can again use normal editing commands. However, no file should EVER by that large! For many reasons it makes sense to break up your files into manageable units no larger than 32,000 characters.

Whenever an error message window is displayed *Ved* will halt until a key is pressed. Note also that errors terminate the operation of macros.

VSplit

Included on the distribution diskette is the program *Vsplit* which will split large files into pieces *Ved* can handle. This utility will not work in a 128K system. It is a packed Basic09 utility so you'll need *RUNB* in memory. To invoke you can either specify a filename:

```
       vsplit ("bigfile")
```

or, if you do not specify a filename, you can enter it at the prompt. *Vsplit* will break the file into files each about 30K long. *Vsplit* attempts to end each new file at a carriage return so that you don't have a sentence split into two files. Each new file will have the original filename with an added section number.

MakeVDefs

This *Basic09* procedure is supplied in both source and I-Code format. This program will take a standard text file with special commands in it and convert it to a *Ved..Defs* file. To get a feel for the syntax we suggest you examine the file *VedDefs.txt*. *MakeVDefs* reads a file consisting of individual lines. The first characters of a line determine what action the program will take. Note that blank lines are ignored, they can be used to make your file more readable.

Comments	Any line starting with a "*" is considered to be a comment line and is ignored.
TABS=	This will set the tabs. Follow the "=" by a list of values representing tab stops. Each value must be from 1 to 80 and separates from the next by a comma.
WRAP=	Set the word-wrap mode. Follow the "=" by either "ON" or "OFF".
OVERSTRIKE=	Set edit mode. Follow the "=" by either "ON" or "OFF".
INDENT	Set the auto-indent mode. Follow the "=" by either "ON" or "OFF".

CASEMATCH=	Set the case-match flag. Follow the "=" by either "ON" or "OFF".
Ox=	Define a two key macro. Note that "x" represents any character from "A" to "Z". Follow the "=" with the text for the macro definition.
x=	Define a single key macro. Here the "x" represents any of the valid one key macro names.

When entering the text for a macro definition you can include both standard text and control characters. Standard text is entered by entering the appropriate characters. For example:

```
      A=Bob van der Poel
```

Will set single key macro "A" to the text "Bob van der Poel".

Control keys can be included by prefixing the name of the key with a "\". The names of the special keys are:

\UP	The Up arrow
\DOWN	The Down arrow
\LEFT	The Left arrow
\RIGHT	The Right arrow
\SHFT	The Shift key
\CTRL	The Control key
\ALT	The Alternate key
\ENTER	The Enter key
\SPACE	The Spacebar
\BREAK	The Break key

These keys can be combined with the "-". For example:

\CTRL-UP	Means hold the CONTROL key and the UP ARROW.
\ALT-F	Means to hold the ALT key and the F key (a *Find*).

Examine the supplied file for more examples.

When you run the program you will be prompted for the name of a text file to convert and the name of *Ved..Defs* file. Pressing just the Enter key in response to the second prompt will cause the program to use the default filename *Ved..Defs*. You can use any valid filename, but *Ved* can only load the file *Ved..Defs*.

<u>VDefsText</u>

This utility, again supplied in *Basic09* source and I-code format, will take an existing *Ved..Defs* file and convert it to a text file. You can then edit this file with *Ved* and add comments or other commands. In reality, this program should be unnecessary since you should always keep the original text file used by

MakeVDefs…but sometimes things get lost. Besides, it is also useful for extracting macros you have created from within *Ved*.

At the prompt you will be asked for the name of a file to convert to text. If you just press the Enter key the program will assume you wish to convert the file *Ved..Defs* in your current data directory. Next enter the name of the text file to use, and the program will do its thing. This program is not well error trapped so anything seriously wrong will cause it to terminate.

The Fine Print

This manual was created using *Ved*. It was printed using the *Vprint* text formatter from Bob van der Poel Software on a Roland PR-1212 printer in proportional and elite print.

All the programs in the package as well as this documentation are protected by copyright. The original purchasers are granted permission to make copies of the programs (but not the documentation) for their own personal use. Neither the programs nor the documentation can be lent to anyone, placed in any library or any type, or given away. The making of illegal copies is prohibited by federal law. Remember, software piracy is theft.

OS-9 is a trademark of Microware System Corporation.

Your comments, suggestions and questions are always welcome at Bob van der Poel Software. But if you want an answer to your query you’ll have to abide by a couple if conditions: First a proof of purchase must be enclosed (frankly, we’re tired of answering questions from people who have stolen our software)—save the receipt. Second, to help offset the high cost of postage and stationary, one dollar (cash, money order or check) must be enclosed.

Thank you for the support you have shown us by purchasing this package. If you’d like to be added to our mailing list for future upgrade and other product information please feel free to write.

Index