DECUS
PROGRAM LIBRARY

DECUS NO.	8-938 (EB)
TITLE	VISTA EDITOR
SUBMITTER	Wally Kalinowski
COMPANY	Aerospace Corp.
DATE	April 1986
SOURCE LANGUAGE	PAGE8

8-838 VISTA EDITOR Version: April 1980

Author: Stewart DeWar

Submitted by: Wally Kalinowski, Aerospace Corp., Los Angeles, CA
Operating System: OS/78, OS/8 Source Language: PAL8 Memory
Required: 12Kw Keywords: Editors

Abstract: VISTA is a full screen editor which allows for scrolling
forward and backward. By means of 'VCM' modules, this
editor can be made to work with any CRT. It supports many
features including:

. String/word search
. Step/iterative replacement
. Status information
. Pikup/putdown, etc.

An updated user manual is supplied (hardcopy only) as well
as the original manual which is on a disk. Also, included
on disk are: HELP.SV,VERSN3.SV,PAGE3.SV,FLIST.SV,BATCH.SV,
HELP.SV,ACID.SV AND DIRECT.SV,CCL.SV. With the exception of
ACID and PAGE8, these programs are enhanced versions of the
originals.

VISTA TEXT EDITOR

PROGRAM REFERENCE MANUAL

March 1, 1980

This manual was generated with DISC's Automatic Cross-referencing and Indexing Document generator program (ACID).

Reformatted and reprinted in August, 1984 by Eugene J. M. Lynch using Xerox Corporation's Word Processing. Electronic Composition and Electronic Printing equipment.

DEWAR INFORMATION SYSTEMS CORPORATION
221 West Lake Street
Oak Park, Illinois 60302

(312) 524-1644

Table of Contents

1. Introduction

VISTA is a new full screen editor that runs on all PDP/8 minicomputers that support a minimum OS/8 configuration. VISTA is an ideal editing program for updating or creating any kind of ASCII source files. One of the main virtues of VISTA is its simplicity--anyone can learn to use VISTA in a matter of minutes. Here are some highlights of VISTA's advanced features:

* *FULL SCREEN EDITING.* The entire CRT screen forms a window into the file that is constantly updated as changes are made to the text. Insertions may be made by character or by line. Deletions can be by character, word, line or block. All changes are displayed as they are made.

* *CURSOR MOVEMENT.* The cursor can be moved up/down/left/right by character, left/right by word, to top left, to bottom right, to next line, to the end of the line.

* *SCROLLING.* The file window can be scrolled backwards and forwards by line or by page (a page being however many lines fit on a particular CRT screen).

* *WORD WRAP.* Vista maintains full wordwrap on the screen: words are not allowed to overrun the right hand margin. At the same time, VISTA is capable of handling files with lines that exceed the screen width without corrupting the original line breaks when the file is closed out.

* *FLEXIBILITY.* Owing to the use of a separate Video Characteristics Module (VCM) to drive the CRT, any CRT currently available can be efficiently supported by VISTA. VCM files provided with VISTA support most of the standard CRTs and new VCM files can be trivially written for other CRTs. The use of a VCM file makes it simple to re-define your own keyboard layout if you should choose. For example on a VT52 terminal, the auxiliary keypad is used to implement various functions such as cursor movement etc. To exchange the functions of two of those keys would require no more than a two instruction patch to the VCM file.

* *HIGH SPEED.* VISTA closes files at extremely high speed: on an RK05 VISTA can close a 200 block file in three seconds. VISTA will automatically maintain the prior version of the file as a .BK file unless overridden with a run-time option.

* *SPECIAL FUNCTIONS.* VISTA includes a string search, a word search and a replacement function. With the latter, there are two modes: step and continuous. In step mode, you can press the REPLACE key to make a replacement, or press the SEARCH key to skip a replacement. In the continuous mode, VISTA proceeds directly to the end of the file making replacements and then displays a count of the number of replacements made. Other special functions include case inversion of a word with a single keystroke, transparent entry of control characters, backspace delete, and high speed movement to the start or end of the file.

- *PICKUP/PUTDOWN.* VISTA can pickup any number of lines of text and put them down elsewhere in a file. Up to 26 named pickup areas are allowed. Pickup areas can also be closed out as OS/8 files and picked up in subsequent edits.

- *STATUS LINE.* A 'hidden' status line invoked by function key displays the name, extension and version of the file being edited along with a count of the amount of space still available for file expansion.

- *HEURISTICS MODE.* In this run-time mode, VISTA turns on various heuristics to speed up the entry of assembler or compiler source programs. For example, text is entered at column 8. If a label delimiter is found, the text is moved into the label field. If a comment delimiter is found, the line is reformatted based upon whether prior text had been entered on the line.

- *VERSION MAINTENANCE.* If the word 'version' appears on line two of a file followed by a number in the form '01.06' VISTA will treat it as a file version number and automatically increment that number every time the file is called up for editing.

- *AUDIT TRAIL.* In conjunction with the VERSION feature, VISTA is capable of automatically appending the current version number to the end of any line that is modified during the current edit. This makes it possible to maintain an automatic record of all changes made to a file.

ACKNOWLEDGEMENTS

Several individuals have made valuable contributions to the structure of VISTA. John Youngquist's ideas on improving the high speed close and providing for high speed positioning by relative block number have been especially useful. Steven Bogolub's Terminal Control Unit Program (TCUP) has also been a source of significant information.

2. Installation Notes

When VISTA is ordered, the following programs will be on the distribution media:

VISTA.SV	The VISTA editing program
VCM0.SV	Video characeristics module for VT-52/VT-78
VCM1.SV	Video characeristics module for VT-100
VCM2.SV	Video characeristics module for VISUAL-200
VCM3.SV	Video characeristics module for BEEHIVE
VCM4.SV	Video characeristics module for HAZELTINE
VCM52.PG	PAGE8 VCM source file for VT-52
VCM100.PG	PAGE8 VCM source file for VT-100
VCM200.PG	PAGE8 VCM source file for VISUAL-200
VCMBEE.PG	PAGE8 VCM source file for BEEHIVE
VCMHAZ.PG	PAGE8 VCM source file for HAZELTINE
VCM52.PA	PAL8 VCM source file for VT-52
VCM100.PA	PAL8 VCM source file for VT-100
VISTA.SD	PAGE8 VISTA Symbol Dictionary file
VISTA.EQ	PAL8 VISTA equivalences

VCM101.PA improved VT100 V1.51 ! —> implemented.

In order to install VISTA, copy over VISTA.SV and the VCMx.SV file that matches the terminal you are using. Then rename that VCMx.SV file to VCM0.SV since VCM0.SV is the default VCM file used when VISTA is called up. If you are installing VISTA and need to re-assemble the VCM source file, also copy the appropriate VCM file and the symbol dictionary. Note that the VCM100 and VCM52 source file are distributed in both PAGE8 and PAL8 source format. Before working on the VCM files, read the VISTA manual carefully, especially the section on creating your own VCM source file.

When the VISTA program is run, VISTA issues a 6107 IOT to see if VISTA is running under a time-share system. If so, VISTA executes a 6047 IOT with an argument list to break on all characters.

NOTE 1: A few terminals have internal timing problems on some editing functions that even differ within serial numbers of the same model. If you run into difficulty using VISTA (especially if certain editing functions do not seem to work properly) contact DISC for assistance. In almost all cases, the problems are quite trivial and can be completely corrected in the VCM file.

NOTE 2: Ideally the video display terminal should be driven as fast as possible. 9600 or 19200 baud is best but 2400 and 4800 are acceptable. 1200 baud is probably the lower limit. VISTA will work at any baud rate but certain operations (especially inserting at the start of a line) start to get very slow at 1200 baud.

3. Using the VISTA Program

To call the VISTA program, issue the following command to OS/8:

```
R VISTA
```

VISTA will respond by typing an '*' to show that the COMMAND DECODER is active. Alternatively, "VISTA" can be patched into the CCL program to support a direct call from the keyboard monitor with the 'EDIT' and 'CREATE' commands. (When modifying CCL, note that VISTA calls the COMMAND DECODER in "special mode").

3.1 Input File

VISTA allows one input file to be specified. No default extension is provided for by VISTA since the COMMAND DECODER is called in special mode. If no input file is specified, an output file must be specified.

3.2 Output File

One output file is allowed. If no output file is specified, the device and filename of the first input file is used for the output file. VISTA does not permit an output file with the extension '.TM' to be used since the 'TM' extension is reserved by VISTA for the creation of the temporary file. For example, the command below:

```
*RKA1:PROG.PA
```

is interpreted by VISTA to mean:

```
*RKA1:PROG.PA<RKA1:PROG.PA
```

VISTA users who are running on tape-based systems should be aware of a feature in VISTA to optimize performance: if a block count is declared with the command decoder square bracket construction, VISTA will use that count as the size of the temporary file for output rather than the actual size of the empty returned by OS/8. Since during backwards scrolling, text is written at the end of the temporary file, this can avoid a substantial amount of tape movement, especially when editing a short file on an almost empty tape. Without the output block count, VISTA would normally move to the extreme end of the tape to write blocks during backward scrolling. As an example of this, assume that we are editing a file called 'ENFORC.PG' which is 150 blocks long. If we are only going to make a few minor editing changes, the new edited version of the file is going to be almost exactly the same length. For the sake of safety, however, it is probably a good idea to allocate at least fifteen or so extra blocks:

```
*DTA0:ENFORC.PG[165]<DTA2:ENFORC.PG
```

3.3 Help File

If the return key is struck to the '*' of the command decoder (or if there are no input or output files), VISTA will output a help file to the console that summarizes all the available commands and run-time options.

3.4 Run Time Options

3.4.1 /A Option

This option enables the 'audit-trail' feature of VISTA. When this option is selected, every statement that is modified during the current editing cycle will be flagged with the current version level of the source program. This option is only effective if the file being edited has a version number on line 2. The comment delimiter and version number are placed starting at the audit trail column number on the current statement. Versions prior to 01.08 displayed the version instantly on the screen. However, version 01.08 and later versions do not append this information to the current line until that line is scrolled off the screen (to improve throughput).

When this option is selected, the bell/buzzer will beep if the cursor moves into a column which will be overwritten by the audit trail information. Note that in almost all cases, the cursor should not be moved into this column because anything typed in that column will be overwritten.

3.4.2 /B Option

This option inhibits the automatic file backup normally performed by VISTA. If there is already a file on the output device with the same name, it will be deleted when the output file file is closed. Normally, such a file would have its extension changed to ".BK" to identify the old file as a backup file.

3.4.3 /C Option

This option is only significant when used in conjunction with the /H option. The /H option normally forces lower case characters into upper case if they appear before the comment delimiter. With the /C option, lower case characters will not be forced into upper case.

3.4.4 /F Option

This option causes VISTA to strip all form feeds in the input source file. After a considerable number of editing passes with other text editing programs such as the OS/8 editor, form feeds and short pages may proliferate. By editing the file with the /F option all prior form feeds will be stripped and ignored. Note that VISTA does not require any form feeds for normal operation. Form feeds present in the original may therefore be deleted or inserted in the same manner as any other special embedded control character.

3.4.5 /H Option

This option turns on certain heuristics for speeding up the entry of assembler or compiler source programs. These heuristics only take effect when text is being entered at the end of the current line. VISTA takes the ASCII codes for the comment field delimiter and the label field delimiter from the VCM file. The precise rules invoked by this option are described in section 10.

3.4.6 /M Option

The /M option implements the correction/deletion flag feature of the ICE editor. A correction/deletion flag is a control/A character which is placed at the start of any line that has been modified. These correction flags are used by PAGE8 (automatic paging macroassembler), ACID (document generator program) and HIBOL (Extended commercial BASIC compiler). Correction flags should only be used with programs that are setup to process these flags properly. Any file with the extension 'PG' (PAGE8), 'AC' (ACID), or 'HB' (HIBOL), will automatically have these flags inserted. With those files, the /M option turns *OFF* the correction/deletion flag feature. In this latter case, the /M option can be used to purge all of the old correction flags from a file. Remember, however, to scroll to the end of the file to get all of the flags purged (Control/O can be used to speed up the operation of scrolling to the end of the file).

With all other file extensions, the correction/deletion flag feature is normally disabled. In that case the /M option turns the correction/deletion flags feature *ON*.

3.4.7 /O Option

If no output file is specified, VISTA automatically defaults the output filename to the input filename. If no output device is specified, VISTA defaults the output device to the input device. In order to edit a file and generate no output, the /O option can be used. When /O is invoked, VISTA will not write anything to the output device. This option might be useful when VISTA is just being used to browse through a file and you intend to abort when done.

3.4.8 /P Option

This option inhibits the High-speed close feature of VISTA. When closing a file, VISTA normally saves a considerable amount of time by padding the output file with nulls (ASCII 200 codes) to align the output file to the nearest block boundary. The remainder of the file is then copied over 15 blocks at a time in image mode. Previously generated nulls are usually stripped out on subsequent edits and are ignored by almost all OS/8 programs. If this "null-padding" is undesirable, the /P option reverts to a slower, character-by-character close. This option is also useful when a very small file is being edited on a frequent basis. In this case the high-speed close is unnecessary and the null-padding may increase the length of the file by a relatively large amount.

3.4.9 /V Option

This option prevents VISTA from incrementing the version number on line 2 of the source file. The original version number is left at its present level.

3.4.10 /W Option

The /W option sets VISTA in wordwrap mode. Normally VISTA is in program edit mode and assumes that all line breaks (CR/LF's) will be explicitly indicated with the return key on the keyboard. With the /W option, however, VISTA runs in wordwrap mode: whenever a word is about to wrap over the right margin, VISTA automatically places a CR/LF on the prior line. If there is room on the next line, the word is merged at the start of that line. If there is not enough room,

a new line is opened up. If the cursor is located in the word at the end of the line, a new line is always opened up.

3.4.11 /Z Option

This option causes VISTA to ignore the ASCII Control/Z code which marks the end of the file. VISTA also ignores the physical end-of-file (*i.e.* based on the number of blocks in the file). This option makes VISTA run in a mode similar to the so-called "SUPER TECO" and allows VISTA to be used as a retrieval utility after a directory crash or inadvertent file deletion. The normal method of using /Z is to use it in conjunction with the command mode 'M' command and the search function. After locating the desired string of text the '*M*' command will mark all lines scrolled off the screen during the search for the end of the file. The *PICKUP* function will then pickup all the text and the '*C*' command will close the file out.

When editing a file with the /Z option, the end of file character (control/Z) is displayed on the screen as a graphic character in the same manner as any other ASCII control character.

3.4.12 /n Option

In this case, '*n*' is a digit from 1 to 9. This option is used to select an alternate VCM file for editing. Normally, when VISTA is brought up, VISTA uses a VCM file with the name 'VCM0.SV'. However, if the command decoder string includes a /2 option for example, VISTA will then look up a VCM file with the name VCM2.SV. Alternate VCM files are typically used to set different characteristics for other display terminals, different delimiters for the label/comment characters or to handle different display modes of a more sophisiticated display terminal.

4. Initial Messages

Before establishing the screen, VISTA will briefly display the following message:

```
VISTA--xx.yy......COPYRIGHT (C), 1980
DEWAR INFORMATION SYSTEMS CORPORATION
```

where 'xx.yy' represents the current level and version number of the VISTA program. Assuming that there are no error or warning conditions pending, VISTA will then read in the first section of the input source file and display it on the screen. Otherwise, one of the messages indicated below will appear:

4.1 Warning Message

```
LESS THAN 10 BLOCKS FOR FILE EXPANSION!
```

This message indicates that there are less than ten extra blocks left for editing. The number of blocks for file expansion is calculated by taking the size of the allocated empty slot for the output file and then subtracting from that the length of the first input file being edited. This number is then decreased by one to account for the possible increase in the length of the file by one block which could be generated by the high-speed close. If the /P option is used, this extra block is not needed (and is not accounted for in the calculation). A value of 0 indicates that if nothing is added to the file, it will just fit. However, we would normally recommend aborting the edit and making more room available on the output device.

Before displaying this message, VISTA will beep the bell/buzzer on the display terminal. This message is displayed for about two seconds. After two seconds, VISTA continues normally to display the screen as described under the normal conditions.

4.2 Warning Message

```
NOT ENOUGH ROOM FOR WHOLE INPUT FILE!
CONTINUE ANYWAY? (Y/N):
```

This message alerts the user to the fact that the empty space allocated for the output file is insufficient to hold the entire input file. Unless a file is being reduced in size, this warning usually means that the user should type 'N' or ↑C to abort since the input file can not be closed out. Attempting to close out the output file when there is not enough room will generate the error message: OUTPUT FILE OVERFLOW. If the lack of space is of no concern, you may still proceed with the edit by typing a 'y'. In that case, VISTA will proceed with the edit as if there was enough room. You should be careful, however, to constantly monitor the status line display to verify that there is room to close out the output file before issuing the close command.

4.3 Other Error Messages

4.3.1 CAN'T OPEN OUTPUT FILE

This message appears if the OS/8 system returns an error on the USR ENTER function. Typical errors would be device full, device write-protected, or read-only device specified for output *etc.*

4.3.2 CAN'T LOAD HANDLER

This message appears if the OS/8 system returns an error when a device handler was loaded by the VISTA program. A typical error would be a non-existent device handler.

4.3.3 INPUT FILE NOT FOUND

This message will appear when VISTA could not find the input file in the OS/8 COMMAND DECODER command string. VISTA returns to the KEYBOARD MONITOR.

5. General Error Messages

While VISTA is running, there are two additional error messages that can appear at any time:

5.1 I/O ERROR! RETRY?

This message appears when an OS/8 device handler signals that an I/O error has occurred. Rather than aborting to OS/8, VISTA allows the user to repeat the operation or to skip over the bad block(s). If the user wishes to abort the edit, ↑C can be typed at this point to return to the OS/8 Keyboard Monitor. If an 'n' is typed, VISTA skips over the bad block(s) and continues as if no error had occurred. Any other character causes the I/O option to be repeated. Another error at this point causes the message to be repeated, so the

I/O function can be tried over and over again if desired. This function of VISTA can be invaluable in trying to recover a vital source file that is on an unreliable medium.

5.2 DEVICE FULL!

This message can occur when VISTA finds no more room left for writing to the output device. The output file is not closed and VISTA returns to the OS/8 Keyboard Monitor. If absolutely vital editing has to be recovered, the user can edit the file immediately before the largest empty space on the output file with the /Z option (with the output file on another device) in order to recover most of the editing. Note that frequent reference to the VISTA status line should eliminate the unexpected appearance of this message.

6. PICKUP/PUTDOWN ERROR

This error message appears in conjunction with the pickup and putdown functions. Among the possible error conditions are:

1. Attempting to pickup more than 384 characters when the output file is on the SYS: device (but see note below).
2. Referencing a non-existent PIKUPx.TM file with the command mode 'L' command.
3. Running out of room on the system device on a pickup function.
4. Running out of room in the PIKUPx.TM file when a pickup is performed immediately after a putdown from a looked-up file.

In any of these cases, recover the screen by first restoring the screen with the restore screen function, followed by the command mode function which will exit command mode and restore the top line.

7. Version Maintenance

If line 2 of the file being edited contains the word 'version' (in upper or lower case) followed by two digits, a period and two digits, VISTA assumes that this is a file version specification. The number will be automatically incremented when the first page of the file is brought to the screen. If the last two digits reach 99, the second digit will be advanced, and the last two digits will be reset to 00. If the /V option is invoked, the version number will be left at its current level. Version maintenance makes it possible to eliminate any confusion as to which version of a source file is the most recent version. This is especially helpful when several different people are editing the same file and when the file is being moved from device to device. It is also useful when there may be several backup copies of a file present.

An example of the start of a HIBOL program file that was using the version feature is shown below:

```
.TITLE   ACCOUNTS RECEIVABLE PROGRAM
.SBTTL   VERSION 01.04
```

Before

Now is the time_for all good men to come to the aid of their country.

After

Now is the time for all good men to come to the aid of their country.

Before

Now is the time_for all good men to come to the aid of their country.

After

Now is the time for all good men to come to the aid of their country.

Before

Now is the time for all good men to come to the aid of their country.

After

Now is the time for all good men to come to the aid of their country.

Before

Now is the time for all good men to come to the aid of their country.

After

Now is the time for all good men to come to the aid of their country.

8. VISTA Command Functions

The following section describes all of the VISTA command functions. The precise key on the keyboard that invokes the function is defined in the VCM file and not in VISTA. Since these functions may be placed in different positions, or on alternate keypads *etc.*, only the function is described here. In each case, the display screen changes are shown. The cursor is represented as an underline.

8.1 Cursor Left

The cursor is moved one position to the left on the screen.

If the cursor is in column 0, the cursor will be placed one position to the right of the last character on the previous line. If the current line is the top line of the display, the display will scroll down one line so that the previous line becomes the top line. If the current line is the first line in the file, the cursor will remain in the home position.

8.2 Cursor Right

The cursor is moved one position to the right on the screen.

If the cursor is already one position past the last character on the line, the cursor will be moved to the start of the following line. If the current line is the last line of the display, the display will scroll up one line so that the following line becomes the bottom line. If the current line is the last line in the file, the cursor will stay at the end of that line.

8.3 Cursor Up

The cursor is moved up one position on the screen.

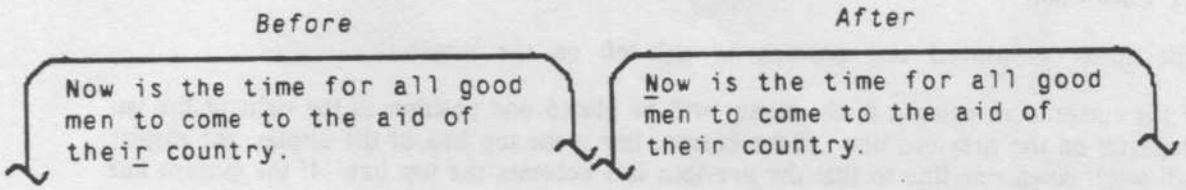
If the current line is the top line of the display, the display will scroll down one line so that the previous line becomes the top line. If the current line is the first line in the file, the cursor will move to the home position.

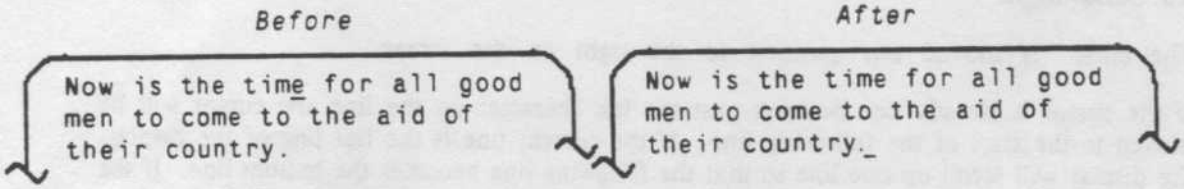
When moving the cursor up or down, VISTA will not allow the cursor to move past the end of the new line. If the new line does not extend as far as the current cursor position on the original line, the cursor will be moved one position past the last character on the new line. However, VISTA will remember that the cursor had to be positioned there because the line was short. As soon as the cursor is moved onto a longer line, VISTA will try and restore the original column that the cursor was in. This 'target columning' feature is only in effect if the cursor up or cursor down keys are struck within about 3/4 of a second. After that time, VISTA will reset the target column back to the current column. What this means is that VISTA will only try and keep in the target column if the cursor up or cursor down key is repeating rapidly.

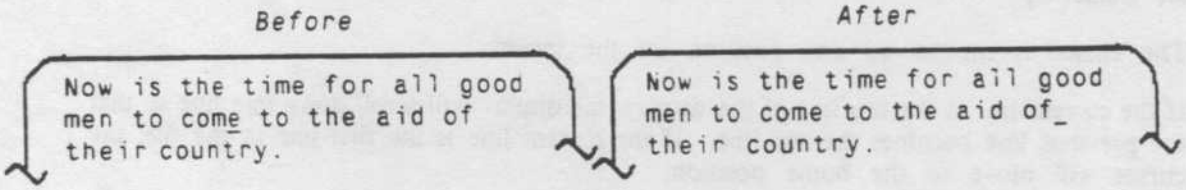
8.4 Cursor Down

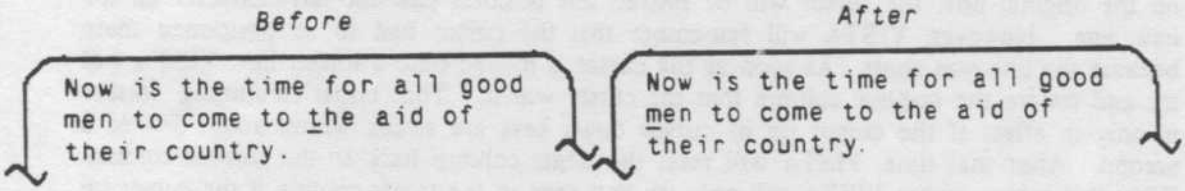
The cursor is moved down one position on the screen.

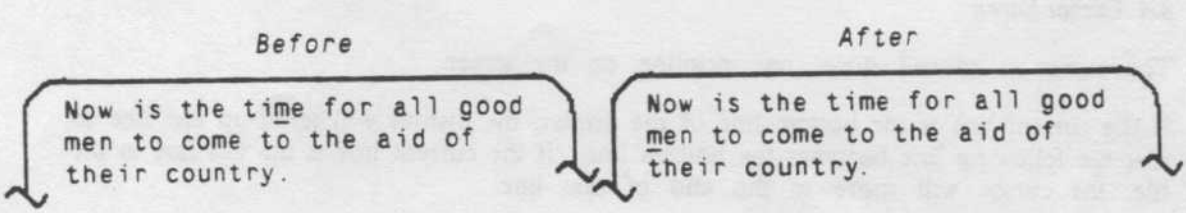
If the current line is the bottom line of the display, the display will scroll up one line so that the following line becomes the bottom line. If the current line is the last line in the file, the cursor will move to the end of that line.


Before
Now is the time for all good men to come to the aid of their country.
After
Now is the time for all good men to come to the aid of their country.


Before
Now is the time for all good men to come to the aid of their country.
After
Now is the time for all good men to come to the aid of their country._


Before
Now is the time for all good men to come to the aid of their country.
After
Now is the time for all good men to come to the aid of_ their country.


Before
Now is the time for all good men to come to the aid of their country.
After
Now is the time for all good men to come to the aid of their country.


Before
Now is the time for all good men to come to the aid of their country.
After
Now is the time for all good men to come to the aid of their country.

When moving the cursor up or down, VISTA will not allow the cursor to move past the end of the new line. If the new line does not extend as far as the current cursor position on the original line, the cursor will be moved one position past the last character on the new line. However, VISTA will remember that the cursor had to be positioned there because the line was short. As soon as the cursor is moved onto a longer line, VISTA will try and restore the original column that the cursor was in. This 'target columning' feature is only in effect if the cursor up or cursor down keys are struck within about 3/4 of a second. After that time, VISTA will reset the target column back to the current column. What this means is that VISTA will only try and keep in the target column if the cursor up or cursor down key is repeating rapidly.

8.5 Home Cursor

The cursor is moved into the home position on the screen. The home position is the top left hand corner of the display: line 0, column 0.

8.6 Cursor Away

The cursor is moved into the away position on the screen. The away position is the bottom right hand corner of the display. Specifically, the cursor will be moved one position to the right of the last character of the last line on the screen.

8.7 Cursor to End of Line

The cursor is moved one position to the right of the last character on the current line.

8.8 Cursor to Start of Line

The cursor is moved to the first character position on the current line: column 0.

8.9 Cursor to Start of Next Line

The cursor is moved to the first character position on the next line: column 0.

If the current line is the bottom line of the display, the display will scroll up one line so that the following line becomes the bottom line. If the current line is the last line in the file, the cursor will remain on the bottom line.

Before

```
Now is the time for all good
men to come to the aid of
their country.
```

After

```
Now is the time for all good
men to come to the aid of
their country.
```

Before

```
Now is the time for all 'good'
men to come to the aid of
their country.
```

After

```
Now is the time for all 'good'
men to come to the aid of
their country.
```

Before

```
Now is the time for all 'good'
men to come to the aid of
their country.
```

After

```
Now is the time for all 'good'
men to come to the aid of
their country.
```

Before

```
TAD   TEMP  /GET VALUE
DCA   FORT  /SAVE FOR LATER
```

After

```
TAD   TEMP  /GET VALUE
DCA   FORT  /SAVE FOR LATER
```

Before

```
TAD   TEMP  /GET VALUE
DCA   FORT  /SAVE FOR LATER
```

After

```
TAD   TEMP  /GET VALUE
DCA   FORT  /SAVE FOR LATER
```

Before

```
TADTEMP  /GET VALUE
DCA   FORT  /SAVE FOR LATER
```

After

```
TAD   TEMP  /GET VALUE
DCA   FORT  /SAVE FOR LATER
```

8.10 Cursor Word Left

If the cursor is at the beginning of a word, it is moved to the beginning of the previous word. In all other cases, the cursor is moved to the beginning of the current word.

With respect to this cursor movement, a 'word' is defined to be a contiguous group of characters consisting of upper or lower case alphabetic characters or the digits 0-9.

8.11 Cursor Word Right

The cursor is moved to the beginning of the following word.

With respect to this cursor movement, a 'word' is defined to be a contiguous group of characters consisting of upper or lower case alphabetic characters or the digits 0-9.

8.12 Cursor Tab Left

The cursor is moved to the nearest tab stop that is immediately to the left of the current cursor position. Tab stops are automatically set every 8 columns starting at column 0. Tab stops subsequently appear at columns 8, 16, 24, 32 *etc.*

If the cursor is in column 0 (left hand margin), the cursor will be moved to the last tab stop on the previous line that matches or precedes the last character position on that line. The display will scroll down if the cursor was in column zero of the top line on the display.

8.13 Cursor Tab Right

The cursor is moved to the nearest tab stop that is immediately to the right of the current cursor position.

There are two special conditions affecting tab right:

If the cursor is at the end of the line a tab code will be appended to the end of the line in order to move the cursor to the next tab stop. If the next tab stop is located at or past the end of the physical line on the screen (as defined in the VCM file), the cursor will not move.

If the cursor is not at the end of the line and insert character mode is on, an ASCII tab code (211 octal) will be inserted at the current cursor position.

Before

```
Now is the time for all good
men to come to the aid of
their country.
```

After scrolling one line

```
men to come to the aid of
their country.
This is the fourth line.
```

Before

```
men to come to the aid of
their country.
This is the fourth line.
```

After scrolling one line

```
Now is the time for all good
men to come to the aid of
their country.
```

8.14 Scroll Up

When the *Scroll Up* key is pressed, the screen is placed in scroll mode. In this mode, the display will scroll continuously until a key is struck (normally, but not necessarily, the same key that invoked the scrolling). When the screen is scrolled one line, the top line is moved into the above screen buffer and every line on the screen is moved up one position. The line waiting immediately below the last line on the display is then moved onto the last line of the display screen. If the line containing the cursor is still on the screen when scroll mode is stopped, the cursor will move to that line. If the original line has left the display, the cursor will be left on the top line of the display, in the original column position, or at the end of that line (if the line is shorter than the original column position).

Continuous scroll mode is automatically terminated when the last line of the file is scrolled onto the screen. If the *Scroll Up* key is struck when the last line of the file is on the screen, the screen will scroll one line and then stop. When the last line of the file reaches the top of the screen, the *Scroll Up* function is ignored. The *Scroll Up* function removes the terminal from insert character mode if insert character mode was in effect when the *Scroll Up* key was pressed.

If enabled in the VCM file, Control/O can be used to turn screen output on and off while scrolling to speed up the scrolling operation (it may be necessary to use the restore screen function afterwards).

8.15 Scroll Down

When the *Scroll Down* key is pressed, the screen is placed in scroll mode. In this mode, the display will scroll continuously until a key is struck (normally, but not necessarily, the same key that invoked the scrolling). When the screen is scrolled one line, the bottom line is moved into the below screen buffer and every line on the screen is moved down one position. The line waiting immediately above the first line on the display is then moved onto the first line of the display screen. If the line containing the cursor is still on the screen when scroll mode is stopped, the cursor will move to that line. If the original line has left the display, the cursor will be left on the bottom line of the display, in the original column position, or at the end of that line (if the line is shorter than the original column position).

Continuous scroll mode is automatically terminated when the first line of the file is scrolled onto the screen.

The *Scroll Down* function removes the terminal from insert character mode if insert character mode was in effect when the scroll key was pressed.

If enabled in the VCM file, Control/O can be used to turn screen output on and off while scrolling to speed up the scrolling operation (it may be necessary to use the restore screen function afterwards).

If the top line of the screen is the first line in the file, the *Scroll Down* function is ignored.

Before

```
men to come to the aid of
their country.
This is the third line.
This is the fourth line.
```

After

```
This is the fifth line.
This is the sixth line.
This is the seventh line.
This is the eighth line.
```

Before

```
This is the fifth line.
This is the sixth line.
This is the seventh_line.
This is the eighth line.
```

After

```
men to come to the aid of
their country.
This is the third line.
This is the fourth line.
```

Before

```
This is the 1024th line.
This is the 1025th line.
This is the 1026th_line.
This is the 1027th line.
```

After

```
Now is the time for all good
men to come to the aid of
their country.
This is the fourth line.
```

Before

```
This is the fifth line.
This is the sixth line.
This is the seventh_line.
This is the eighth line.
```

After

```
This is the 2850th line.
This is the 2851st line.
This is the 2852nd line.
This is the last line.
```

8.16 Page Up

When the *Page Up* key is pressed, all lines on the screen are moved into the above screen buffer. As many lines as are available on the screen are then moved from the below screen buffer on to the screen. The *Page Up* function is functionally equivalent to scrolling until the last line that was on the screen has moved into the above screen buffer. *Page Up* is faster than scrolling because the screen is blanked while the lines are moved. The screen is then restored at the termination of the *Page Up* function and the cursor is left in the home position.

The *Page Up* function automatically removes the terminal from insert character mode.

8.17 Page Down

When the *Page Down* key is pressed, all lines on the screen are moved into the below screen buffer. As many lines as are available on the screen are then moved from the above screen buffer on to the screen. The *Page Down* function is functionally equivalent to scrolling until the first line that was on the screen has moved into the below screen buffer. *Page Down* is faster than scrolling because the screen is blanked while the lines are moved. The cursor is then at the home position.

The *Page Down* function automatically removes the terminal from insert character mode.

8.18 First Page

When the *First Page* function is called for, VISTA moves to the very first page in the file and then displays that page. The cursor is moved to the home position.

Version 01.10 and subsequent versions move to the first page at a very high rate of speed. In moving to the start of the file VISTA inserts on average, a single ASCII null code (200 octal) into the output file for alignment purposes. If the `/P` option was invoked in the command string, VISTA will avoid the insertion of the null code and substitute the slower file movement of versions 01.09 and prior versions of VISTA.

8.19 Last Page

When the *Last Page* function is called for, VISTA moves to the very last page in the file and then displays that page. The cursor is moved to the home position.

Version 01.10 and subsequent versions move to the last page at a very high rate of speed. In moving to the start of the file VISTA inserts on average, a single ASCII null code (200 octal) into the output file for alignment purposes. If the `/P` option was invoked in the command string, VISTA will avoid the insertion of the null code and substitute the slower file movement of versions 01.09 and prior versions of VISTA.

The display is positioned so that the very last line of the file is placed on the last line of the display screen.

Before

Now is the time for all gofod
men to come to the aid of
their country.

After

Now is the time for all good
men to come to the aid of
their country.

Before

Now is the time for all good
men to come to the aid of
thxeir country.

After

Now is the time for all good
men to come to the aid of
their country.

Before

Now is the time for all good
men to come to the aid of
their country.

After

Now is the for all good
men to come to the aid of
their country.

Before

Now is the 'half-time' for all
men to come to the aid of
their country.

After

Now is the for all
men to come to the aid of
their country.

Before

Now is the 'half-time' for all
men to come to the aid of
their country.

After

Now is the 'h-time' for all
men to come to the aid of
their country.

Before

```
LABEL TAD   TEMP  /GET COUNT
      DCA   CNTR
```

After Delete to End of Word

```
      TAD   TEMP  /GET COUNT
      DCA   CNTR
```

8.20 Delete Character

The character that the cursor is pointing to is deleted from the screen. The remainder of the line is reformatted to ensure that any subsequent tab stops on the line are correctly displayed. After deletion of the character the cursor points to the character that was immediately to the right of the character that was deleted. If the cursor was one character position to the right of the last character on the line when the delete character key was struck, the delete character key functions as a backspace delete as described in section 8.21.

8.21 Backspace Delete Character

The cursor is first moved one position to the left. If necessary, the cursor will be moved onto the prior line and the display will scroll. The character that the cursor is then pointing to is deleted from the screen. The remainder of the line is reformatted to ensure that any subsequent tab stops on the line are correctly displayed. After the backspace delete, the cursor still points to the same character.

The backspace deletion function is especially useful for making corrections when in insert character mode, because you cannot move the cursor left and overstrike any errors. Neither the delete character nor backspace delete functions cause a line to be removed from the screen, since if the line is completely blank, the cursor will be moved onto the previous line.

8.22 Delete Word (Field)

All characters surrounding the cursor and bounded at either side by a character whose ASCII code is less than 241 (esp. space and tab) are deleted. The bounding character on the left is retained, while the bounding character on the right is deleted along with the remainder of the word (field). If the cursor is located at the end of the line or on a word delimiter (something other than an upper/lower case letter or digit), the cursor will first be moved to the left. Note that the definition of 'word' in this function is slightly different from that of the word cursor movement commands and the delete to end of word function. In editing text, the definitions are normally the same since most 'words' in text are bounded by spaces. However, in source programs the definitions are quite different (for example, a long expression with no spaces or tabs).

8.23 Delete to End of Word

All characters in the current word from the cursor position to the end of that word are deleted from the screen. In this context, a word is defined to be a contiguous group of upper/lower case alphabetic characters or the digits 0-9. Note that this definition is different from the definition in the delete word (field) function. After deletion to the end of the word, the cursor is positioned to the start of the following word. If the cursor is not located on a word (for example, the cursor is on the space between words), the delete to end of word function will be ignored. If the cursor is one character position to the right of the last character on the line, the cursor is simply moved to the start of the first word on the next line.

Delete to end of word is especially useful for editing program source files. The example shows how a label is removed in a single keystroke.

Before

```
Now is the time for all good
men to come to the aid of
their country.
```

After

```
Now is the time for all good
their country.
This is the fourth line.
```

Before

```
Now is the time for all good
men to come to the aid of
their country.
```

After

```
Now is the time for all good
men to come to t_
their country.
```

Before

```
Now is the time_
for all good
men to come to the aid of
```

After 2 End of Line Deletes

```
Now is the time$
for all good$
men to come to the aid of_
```

Before

```
Now is the time$
for all good$
men to come to the aid of_
```

After Scrolling Back to Screen

```
Now is the time for all good
men to come to the aid of
their country.
```

Before

```
Now is the time for all good
men to come to the aid of
their country.
```

After

```
This is the fourth line.
This is the fifth line.
This is the sixth line.
```

Before

```
NUMBER OF REPLACEMENTS MADE:85
```

After

```
Now is the time for all good
men to come to the aid of
their country.
```

8.24 Delete Line

The *Delete Line* function has three different functions depending upon the position of the cursor when the *Delete Line* function was invoked.

8.24.1 Line Delete in Column 0

If the cursor is in column 0, the entire line is removed from the screen. If the line being deleted was not the last line of the file, all subsequent lines are moved up one position on the screen and the next line waiting below the screen is moved onto the screen.

8.24.2 Line Delete in Middle of Line

If the cursor is in the middle of the line (not in column 0 and not one character position to the right of the last character on the line), all characters from the cursor position to the end of the line are deleted.

8.24.3 Line Delete at End of Line

If the cursor is one character position to the right of the last character on the line, the current line is marked to indicate that it should be merged with the following line when that line is scrolled off the screen. The cursor is then moved to the end of the following line. The display will scroll up if necessary. When merging two lines, VISTA will supply a blank to separate the two lines rather than the normal CR/LF. The current line is flagged with a special 'no end-of-line' graphic character by the VCM file. This special flagged status can be removed with the *Unmark Line* function.

In this example, the '$' represents the special graphic character used to flag a line that is to be merged. After the flagged two lines are scrolled off the screen, they will be merged into a single line (program edit mode) or a new line break will be computed for the line (wordwrap mode).

In wordwrap mode, after a page down and page up function, we would probably see the screen shown in the example.

8.25 Delete Screen

All lines on the screen are deleted. Lines waiting to appear on the screen are then scrolled onto the screen until the screen is full again.

8.26 Restore Screen

The restore screen causes the screen to be cleared and then restored from the image of the screen that is retained in memory. The cursor is left in the home position. Restoration of the screen would be performed under any of the following circumstances:

1. At the termination of a continuous replacement function after the count of replacements made has been displayed.
2. After an error message has blanked the screen (such as PICKUP/PUTDOWN ERROR, *etc.*).

Before

Now is the time for all good
men to come to the aid of
their country.

After

Now is the time for all good
men to come to the aid of
their country.

Before

Now is the time for all good
men to come to the aid of
ther country.

After striking the letter "i"

Now is the time for all good
men to come to the aid of
their country.

Before

Now is the time for all good
men to come to the aid of
their country.

After

Now is the time for all good
men to come to the aid of

3. If the terminal was accidentally powered off, switched into local mode, or if the terminal 'hiccups' for some reason and it is believed that the display on the screen does not match the display in VISTA's memory. In the latter case, a terminal that hiccups frequently probably has an internal timing problem (most likely on delete or insert line). In that case the VCM file should probably have some more nulls inserted after these functions.

4. To restore the screen after a display modified lines function has been performed.

8.27 Display Modified Lines

This function causes all lines on the screen to be re-displayed as in the restore screen function. However, all lines which have been modified will be flagged visually on the screen.

The mode of flagging depends upon the VCM file and the type of terminal in use. Most terminals have some form of video enhancement (reverse video, half/full intensity, underline *etc.*) in which case the entire line is displayed in this video mode. If no special video enhancement is available, a special flag character of some kind will be displayed at the start of each modified line.

If correction/deletion flags are being processed, VISTA will always 'remember' which lines have been modified even after they have been scrolled off and then scrolled back onto the screen. In all other cases VISTA will only remember which lines have been modified if they have not left the screen since being modified.

In the example, the modified line was the second line which has been flagged by being underlined. Note that when flagging lines, if a line is deleted, the following line will be flagged.

NOTE: If the VCM file in use flags lines with a special character (rather than a video enhancement), all characters on the screen will be displaced by one position from their true position. Attempting to edit lines flagged in this manner will cause problems and should therefore be avoided.

8.28 Insert Character Mode

The *Insert Character* function toggles the current state of the insert character mode flag. If insert character mode was on, it is turned off. If insert character mode was off, it is turned on. When insert character mode is being turned on, the terminal will beep to alert the user that insert character mode is now on.

When the terminal is in insert character mode, all characters struck on the keyboard will be inserted at the current cursor position rather than overstriking the character that was under the cursor.

8.29 Insert Line

The *Insert Line* function opens up a blank line above the line on which the cursor was located. The bottom line of the display is moved into the below screen buffer and all lines from the line containing the cursor to the last line on the screen are moved down one line. The physical line which originally contained the cursor is then cleared and the cursor is placed at column 0 on that line.

Before

Now is the time for all good
men to come to the aid of
their country.

After

Now is the time for all good
_

Before

Now is the time for all good
men to come to the aid of
their country.

After "their"; Caps Lock on

Now is the time for all good
men to come to the aid of
THEIR_country.

Before

Now is the time for all good
men to come to the aid of
_their country.

After transparent key then FF

Now is the time for all good
men to come to the aid of
$their country.

Before

Now is the time for all good
men to come to the aid of
TheIR country.

After

Now is the time for all good
men to come to the aid of
THEir country.

Before

Now is the time for all good
men to come to the aid of
their country.

After

Now is the time for all good
men to come to the aid of
their country.

8.30 Insert Block

The *Insert Block* function opens up all lines to the end of the screen from the line in which the cursor was located. This function is especially useful when a large amount of text is going to entered on a terminal that lacks a native insert line function. After entry of all the text, the *Scroll Up/Down* functions can be used to close up any unused blank lines at the bottom of the screen.

8.31 Caps Lock

The *Caps Lock* function inverts the current setting of the caps lock mode. If caps lock mode was on, it is turned off. If caps lock mode was off, it is turned on. When caps lock mode is being turned on, the terminal will beep to alert the user that caps lock mode is now on.

When the terminal is in caps lock mode, all lower case alphabetic characters struck on the keyboard will be forced to appear in upper case. If caps lock mode is off, characters will appear in whatever mode they were typed in from the terminal.

8.32 Transparent Character Entry

When this character is struck, VISTA will read the next character from the keyboard 'transparently'. In other words, VISTA will not process the character normally. The character read will then be stored at the current cursor position as if it was a normal character. If the character is a non-displayable character (control code or rubout code), a special graphic character will appear on the screen for display purposes. The precise ASCII code of the character can always be determined by placing the cursor on the character and pressing the *Command Mode* function key. This will display the ASCII character code in octal within angle brackets. A common use for this function is to insert special characters such as form feeds.

In this case, the '$' is used to represent the special graphic placed on the screen. The precise graphic character used is a function of the VCM file and the display terminal.

8.33 Invert Case to End of Word

The *Invert-Case-to-End-of-Word* function inverts the upper/lower case characteristic of each alphabetic character from the current cursor position to the end of the word that originally contained the cursor. After inversion of the case of each character, the cursor is moved to the start of the following word via the *Cursor Word Right* function.

8.34 Mark Line

The mark line function causes the current line containing the cursor to be marked for a subsequent pickup operation. The marked line will be flagged either by a video enhancement (such as re\verse video, half intensity, underline etc.) or by a special flag character at the start of the line (this is a function of the VCM file and the display terminal). After the current line is marked, the cursor is moved to the beginning of the following line. If the marked line was the last line on the display, a scroll up operation will be performed. If the current line had been flagged as having been modified or for merging with the following line (by a delete-end-of-line function at the end of a line), these flags are removed. In the example, the marked line (the second line on the screen) has been flagged by a video enhancement (underlining).

Before

Now is the time for all good
men to come to the aid of
their country.

After

Now is the time for all good
men to come to the aid of
their country.

Before

Now is the time for all good
men to come to the aid of
their country.
This is the fourth line

After

Now is the time for all good
their country.
This is the fifth line
This is the sixth line

NOTE: Marked lines which are scrolled off the screen are flagged with a special control character at the beginning of the line. If the file were closed without executing a *Pickup* function, illegal control characters would appear in the file.

NOTE: If the VCM file in use flags lines with a special character (rather than a video enhancement), all characters on the screen will be displaced by one position from their true position. Attempting to edit lines flagged in this manner will cause problems and should therefore be avoided.

8.35 Unmark Line

The *Unmark Line* function unconditionally removes the special status caused by a previous mark-line operation (if any). The unmark line function also removes any possible merge-line status set by a prior delete-end-of-line operation at the end of a line. After the current line status is cleared, the cursor is moved to the beginning of the next line. If the unmarked line is the last line of the display, the screen will scroll up one line.

8.36 Pickup Lines

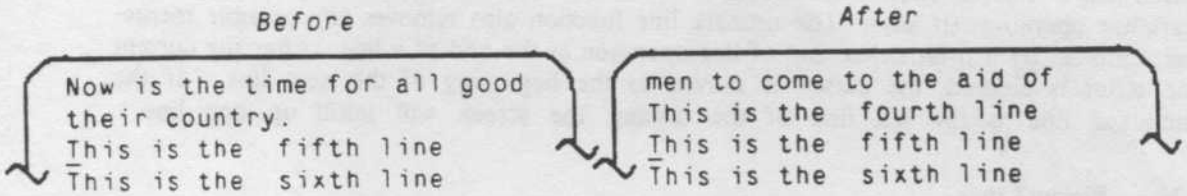
The *Pickup Lines* function locates all lines in the file that have been marked with prior *Mark Line* functions. Each marked line is then stored into a pickup area which starts in memory and moves out onto the system device if room is needed. As each line is moved, it is deleted from the screen. If all lines to be picked up are currently being displayed on the screen, the pickup operation can be observed as it progresses. However, if one or more lines have been scrolled off the display, the display is blanked and will be restored at the termination of the pickup operation. Marked lines need not be contiguously stored. However, all picked up lines will be merged into a single pickup block.

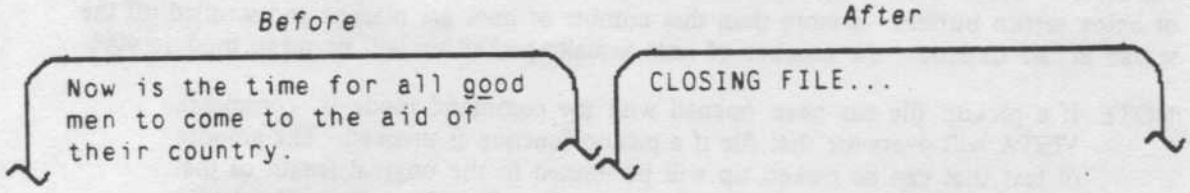
If the number of characters to be picked up fits in less than one OS/8 block (384 characters), the pickup and subsequent putdown operation takes place purely in memory. However, if that 384 character limit is exceeded, VISTA opens up a temporary file on the system device (SYS:) with the name 'PIKUPA.TM'. If the output file is being stored on the system device, the pickup operation is necessarily limited to operation within memory only and any attempt to move out onto the system device will cause a PICKUP/PUTDOWN ERROR message to appear (but see NOTE below).

When the *Pickup Lines* function is invoked, the position of the cursor is of no significance. At the end of the operation, the cursor will be located immediately after the last picked up line. Whenever the *Pickup Lines* function is invoked, the prior picked up text (if any) is always overwritten. Note that the 'C' function in command mode makes it possible to convert the temporary file into a permanent file and to then open up a new temporary file with the name 'PIKUPB.TM' *etc.* for a subsequent pickup operation.

VISTA only keeps track of up 4095 lines scrolled off the screen into either the above screen or below screen buffers. If more than that number of lines are marked and scrolled off the screen in one direction, the number of lines actually picked up will be taken modulo 4096.

NOTE: If a pickup file has been opened with the command mode 'L' command, VISTA will overwrite that file if a pickup function is invoked. The amount of text that can be picked up will be limited to the original length of that pickup file. To avoid overwriting the pickup file, the pickup file should first be closed with a command mode 'C' command.


Before
Now is the time for all good
their country.
This is the fifth line
This is the sixth line
After
men to come to the aid of
This is the fourth line
This is the fifth line
This is the sixth line


Before
Now is the time for all good
men to come to the aid of
their country.
After
CLOSING FILE...

The ability to overwrite a pickup file makes it possible to perform large pickup operations even when the output file is on the system device. This is handled by creating a file called, say PIKUPZ.TM on SYS: with a suitably large length to cover any anticipated pickup operations (20 blocks or so). Prior to the first pickup, the command mode command: 'LZ' will then select this file for the pickup operation and the PICKUP/PUTDOWN ERROR condition will be avoided.

8.37 Putdown Lines

The *Putdown Lines* function puts down all the lines that were saved in the temporary pickup area by the most recent *Pickup Lines* function. As each line is about to be put down, VISTA performs an *Insert Line* function at the current cursor position which opens up a blank line. The first or next line from the pickup area is then copied into the blank line that was opened up. After copying the line, the cursor is then moved down one line. During the putdown operation, if any key is struck on the terminal, the putdown operation is immediately suspended. A subsequent putdown operation would then continue from where the prior operation left off. Lines are displayed on the screen as they are being put down.

If a large number of lines are being putdown the CONTROL/O function can be used to blank the screen. If the screen is blanked, VISTA will automatically restore the screen at the end of the putdown function. Users with terminals that lack a native insert line function may also find this useful since the putdown operation is noticeably slower on such terminals (since the insert line operation requires a partial rewrite of the screen).

There is no limit to the number of lines which can be putdown by this function. Pickup files are written as standard OS/8 ASCII files. For a large merge operation, it is perfectly reasonable to rename a file to PIKUPA.TM before editing and then use the VISTA 'L' (Lookup) command to look up the file, followed by a putdown operation to do the insertion.

After a putdown operation, all internal pointers are reset in VISTA so that if another putdown operation is invoked, the same group of lines will be putdown again. This is especially useful for replicating blocks of text at very high speed.

Remember that any subsequent pickup operation will overwrite whatever was last put down. For this reason, it is good practice to issue a command mode 'C' command if you plan to issue another putdown at a later time.

8.38 Close File

The *Close File* function ends the edit cycle. The file is closed out on the output device with all changes made during the current edit cycle. If there is already a file on the output device with the same name and extension as output file specification to the command decoder when VISTA was called, the extension of that file is changed to .BK to identify it as a backup file. If the /B option was invoked, the file is deleted from the disk rather than having its extension changed to .BK.

If less than 3 pages of space were needed for device handlers, and if the /P option was not called for, VISTA will perform a 'high-speed-close' operation which involves null-padding of the output file to greatly reduce the time needed to close out the file. As soon as the close function has been invoked, VISTA blanks the screen and places the message CLOSING FILE... on the top line.

Before

```
Now is the time for all good
men to come to the aid of
their country.
This is the  fourth line
```

After, on a subsequent edit

```
Now is the time for all good
men to come to the aid of
```

Before, in text mode

```
Now is the time for all good
men to come to the aid of
their country.
This is the  fourth line
```

After entering command mode

```
_  TESTA .AC-03.14 99+ <344>
men to come to the aid of
their country.
This is the  fourth line
```

Before, in command mode

```
_  TESTA .AC-03.14 99+ <344>
men to come to the aid of
their country.
This is the  fourth line
```

After returning to text mode

```
Now is the time for all good
men to come to the aid of
their country.
This is the  fourth line
```

Before

```
Now is the time for all good
men to come to the aid of
their country.
This is the  fourth line
```

After

```
Now is the time for all good
men to come to
the aid of
their country.
```

If the OS/8 system reports an error during the close operation (other than a device handler error), the message CLOSE ERROR will appear and VISTA will return directly to OS/8.

8.39 Truncate File

The *Truncate File* function ends the edit cycle. The file is truncated at the line containing the cursor: the line above the line containing the cursor will become the last line of the new file being created by this edit cycle. After truncating the file, all other conditions are identical to those of the normal *Close File* function.

8.40 Return to OS/8 Monitor

This function causes an immediate return to the OS/8 keyboard monitor by passing control directly to 7600 in field 0. The screen may not be blanked before exiting, and the dot from the keyboard monitor will appear. Files are not closed. Before returning to 7600, however, VISTA does issue an exit function call to the VCM module. Some VCM files use this call to reset the terminal, and to erase the screen (the VT-100 terminal, for example, is reset back into VT-52 mode for OS/8 usage). This reset operation is *not* performed if CONTROL/C (the normal ASCII code for this function) is detected by an OS/8 device handler.

8.41 Enter Command Mode

If the terminal was in text mode, this function causes the terminal to enter command mode. The top line on the screen is saved and a status line is displayed showing the filename, extension, version number (if any), number of blocks available for file expansion, and the octal code of the character under the cursor when the *Enter Command Mode* function was invoked. At this point, any of several command mode functions can be executed. These are described in section 9.

If the terminal was already in command mode, the top line is restored and the original position of the cursor (when the terminal entered command mode) is restored. Any partial command typed in on the top line is discarded and ignored.

In this example, the file being edited is TESTA.AC, with a version number of 03.14. The 99+ indicates that there are at least 100 blocks available for file expansion. The <344> is the octal ASCII code for the lower case letter 'd'.

8.42 Return Function

This function splits a line into two separate lines. All characters to the left of the cursor are left on the current line intact. All characters to the right of the cursor are saved for a moment. An insert line operation is performed on the line immediately below the current line. The saved characters are then copied to that line. Note that if the *Return* code is struck at the end of the line, there are no characters to the right of the cursor, and the net result is that a blank line is automatically opened up below the current line. The character under the cursor is deleted, because the cursor will normally be on a space when the *Return* key is struck. If the cursor is located on a TAB character, the *Return* function is not allowed and the bell/buzzer will beep instead.

Before (search string is "the")

```
Now is the time for all good
men to come to the_aid of
their country.
This is the fourth line
```

After

```
Now is the time for all good
men to come to the aid of
their country.
This is the fourth line
```

Before (search string is "the")

```
Now is the time for all good
men to come to the_aid of
their country.
This is the fourth line
```

After

```
Now is the time for all good
men to come to the aid of
their country.
This is the fourth line
```

Before (replace string is "a")

```
Now is the time for all good
men to come to the aid of
their country.
This is the fourth line
```

After

```
Now is a time for all good
men to come to the aid of
their country.
This is the fourth line
```

Before (replace string is "a")

```
Now is the time for all good
men to come to the aid of
their country.
This is the fourth line
```

After restoring screen

```
Now is a time for all good
men to come to a aid of
their country.
This is a fourth line
```

8.43 Search Function

The *Search* function searches for a string of text that matches the most recent search string specified in a command mode 's' or 'w' command. The search begins at the current cursor position. If a matching string is found on the screen, the cursor is immediately moved to first character of the string that matched the search string. If a match is not found on the screen, the screen is blanked and the search continues. If a match is then found, the screen is restored with the line containing the matching string positioned in the middle of the screen. If no match is found, the screen is restored (which will be the end of the file) and the bell/buzzer will beep to signify that the search failed.

If the *Search* function is invoked when no search string has ever been entered, the search string is treated as a null string which always matches.

The *Search* function can be aborted at any time by striking any key on the keyboard. The struck key is discarded and the screen will be restored at whatever point in the file the *Search* function had reached when the key was struck.

The second example takes the same '**Before**' screen as above, but assumes that the search string was entered with the 'w' command for a word search.

8.44 Step Replacement Function

The *Step Replacement* function first tests to see if the cursor is located at the start of a string which matches the search string specified by the most recent *Search* function. If so, that string of characters is deleted. The replacement string specified in the most recent command mode 'r' command is then copied into the line to replace the deleted string. After this an immediate search is initiated to find the next occurrence of the search string.

If the replacement string is null, the *Step Replacement* function operates as a string delete function, since the replacement string is null. If the *Step Replacement* function key is struck and the search string does not match the string at the cursor position, the bell/buzzer will beep and the replacement function is ignored.

If the replacement string causes the current line to exceed the screen width, the entire screen is reformatted to allow for splitting the line into two separate lines. The line splitting is performed internally by the wordwrap routines, so that a line break will be forced at a space if one exists on the line being reformatted.

8.45 Continuous Replacement Function

The *Continuous Replacement* function is identical to the *Step Replacement* function except that replacements are performed automatically without pausing until the end of the file is reached. When the end of the file is reached, the screen is cleared, and a count of the number of replacements is displayed on the screen. A count of 99+ indicates that 100 or more replacements were performed. The screen can be restored by the *Restore Screen* function, scrolling, paging or by using the *First/Last Page* functions.

9. Command Mode Functions

All of the following commands are typed in on the hidden status line at the top of the screen. The *Command Mode* function must have been invoked first in order to enter these commands. Also the full row width is available and it is perfectly all right to type over the status line information on the screen. All commands are ended with the RETURN key. If the *Command Mode* function key is

hit on the top line while in command mode, the current command is ignored and VISTA restores the top line and original cursor position.

When the terminal is in command mode. Most of the VISTA editing functions are disabled. The only available editing functions are:

Cursor Left
Cursor Right
Home Cursor
Cursor to End of Line
Word Left
Word Right
Delete to End of Line (but not in home position)
Backspace Delete (functions as *Cursor Left*)

All other editing corrections must be made by overstriking. The command line ends at the cursor position when the RETURN key is struck. Any characters to the right of the cursor when the RETURN key is struck are ignored. Note that insert mode is not available on this command line. When in command mode, VISTA will not allow the cursor to leave the command line.

9.1 B - Block Number Positioning

`Bnnnn` This command is not available in version 01.09 and prior versions of VISTA. In versions 01.10 and later versions, this command causes VISTA to position the screen window so that the text contained in relative block '*nnnn*' (decimal) of the input file is displayed at the top of the screen. The first block in the file is considered to be relative block zero. Positioning by block number uses the same algorithm as the *First/Last Page* functions and on the average inserts a single ASCII null code (200 octal) into the output file. Even with a 250 block file, the worst case positioning command will take about three seconds on an RK05 type disc. SPR's are available for the PAGE8 macroassembler, ACID document generator and HIBOL compiler to list relative block numbers instead of the usual page/line numbers.

In determining the relative block number, if the relative block number is larger than the current input file block number, the specified block number will be the relative block number in the original input file. If the block number is smaller, the relative block number is taken from the start of the new temporary file which is being created during the edit cycle.

9.2 C - Close Current Pickup File

`C` Causes the current pickup area which is normally a temporary file on the `SYSTEM` device to be closed out and made into a permanent file. A new pickup area will be opened for the next pickup operation. For example, the first time the 'C' command is issued, a file with the filename `PIKUPA.TM` will be closed out on the system device. A subsequent pickup command will open a file with the name `PIKUPB.TM`. If the most recent putdown operation was from a file that was looked up with the 'L' command, VISTA simply opens up the next sequential pickup area.

9.3 L - Lookup Pickup File

`Lx` Causes VISTA to look up a pickup file with the name: `PIKUPx.TM`, where '*x*' is the letter specified immediately after the '*L*'. A subsequent putdown function will put down text from this pickup file. A cautionary note: after looking up a pickup file, be careful about doing a new pickup operation since VISTA will overwrite the currently selected pickup file. If you close the pickup file with a 'C' command that will delete the pickup file and allow VISTA to open as large a temporary pickup file as can be opened up.

9.4 M - Mark All Lines

`M` This command causes all lines scrolled off the screen (in either direction) to be flagged for a subsequent pickup operation. It is especially useful when a very large number of lines are to be moved. In addition to the normal scroll operations, lines will be marked on *Page Up/Down* and on the *Search* function. This condition is automatically reset at the termination of the pickup operation.

9.5 S - Search Command

`Stextstring` This command causes VISTA to store the search string '*text string*' into the string search buffer. After storing the string, VISTA restores the top line and cursor position and then invokes a search command function to find the string. The search string can be in either upper or lower case. Note that for the search, VISTA ignores differences in case shift. The first significant character of the search string *immediately* follows the letter 's' of the command (as shown in the example here). Leading blanks are significant, so that the command 's *text*' would search for a five character string beginning with a blank.

9.6 W - Word Search Command

`Wtextstring` The 'w' command is identical to the 's' command except that VISTA must find a word delimiter at either end of the string of characters. A word delimiter is any character other than the letters (both upper and lower case) and the digits 0-9. In addition, the start and end of a line are considered to be word delimiters. As with the 's' command, as soon as the RETURN key is pressed, VISTA will restore the top line and start searching for the string of text. Note that the definition of a word for the word search command is slightly different from the definition used in the word cursor movement and delete functions.

9.7 R - Replacement String

`Rtextstring` The 'r' command defines the replacement string for the step replacement and continuous replacement functions. Whenever a matching search string has been found, the replacement functions will utilize the most recent string of text defined by an 'r' command. If no 'r' command has been issued, the replacement string is defined to be a null string (a null string can also be set by typing a RETURN key immediately after the 'r' of the command). If the replacement string is null, the replacement functions

Before

```
TAD   TEMP  / GET RESULT
DCA   ORG7  / SAVE IT_
```

After pressing the RETURN key

```
TAD   TEMP  / GET RESULT
DCA   ORG7  / SAVE IT
_
```

Before

```
TAD   TEMP  / GET RESULT
TAG_
```

After typing a comma

```
TAD   TEMP  / GET RESULT
TAG.  _
```

Before

```
TAD   TEMP  / GET RESULT
DCA   ORG7  / SAVE IT
_
```

After typing comment delimiter

```
TAD   TEMP  / GET RESULT
DCA   ORG7  / SAVE IT
/ _
```

Before

```
TAD   TEMP  / GET RESULT
DCA   ORG7_
```

After typing comment delimiter

```
TAD   TEMP  / GET RESULT
DCA   ORG7  / _
```

operate as deletion commands (the searched for string is deleted, since the replacement string is null).

As with the 's' and 'w' commands, the replacement string starts immediately after the letter 'r'. Upper and lower case *are* significant in the replacement string: when the replacement function is invoked, the upper/lower case attributes of the original string will be retained.

10. Entry Mode Heuristics

The entry mode heuristics are enabled with the /H run-time option. This option is only effective when typing characters at the end of a line. The single exception to the end of line rule is the upper/lower case folding switch which is in effect at any point within a line. The heuristics are designed to speed up the entry of assembler/compiler source programs by providing automatic field formatting. In the examples that follow, we shall assume that the comment field delimiter is a slash and the label field delimiter is a comma (as would be the case with DEC's PAL8 assembler). The heuristics that are used include the following:

10.1 Start of Line

Whenever the RETURN key is pressed, the new line will start with a tab code so that the cursor is positioned at column 8. The upper/lower case folding switch is also set to upper case so that all instructions will be entered in upper case. The positioning of the cursor in column 8 also takes place automatically after an insert line or insert block function.

10.2 Label Field Delimiter

If the character which marks the end of the label field (as defined in the VCM file) is struck between columns 8 and 15, the tabulation code in column zero is deleted so that columns 8 through 15 now appear in column 0 through 7. The label field delimiter is then followed by a tab code so that the cursor is positioned at column 8. Insert character mode is unconditionally turned off by this function.

10.3 Comment Field Delimiter in Column 8

If the comment field delimiter (as defined in the VCM file) is struck in column 8, VISTA assumes that a full comment line will follow. In that case, The comment field delimiter is moved to column 0 and a space is placed in column 1. The cursor is then positioned in column 2. The upper/lower case folding switch is also set to lower case which will permit lower case characters to be entered in the comment field (providing that the terminal can generate lower case characters and the caps lock switch is off).

10.4 Comment Field Delimiter Not in Column 8

If the comment field delimiter is struck between columns 9 and 31, VISTA will insert tab codes until column 32 is reached. If the cursor is already at or past column 32, this step is omitted. After reaching column 32, VISTA inserts the comment field delimiter followed by a space and then leaves the cursor at column 34. The upper/lower case folding switch is also set to lower case which will permit lower case characters to be entered in the comment field (providing that the terminal can generate lower case characters and the caps lock switch is off).

If the comment delimiter has to be entered in a field other than the comment field, you can type a space followed by the next character, and then move the cursor back to strike the comment delimiter (the heuristics are turned off unless the cursor is at the end of the line). This procedure would have to be used to enter a statement such as:

```
TAD   ("/      / GET SLASH CHARACTER
```

11. VISTA Memory Map

Field 0	00000-05777	VISTA editor program routines
	06000-06577	VCM file area
	06600-07177	Device handler region
	07200-07377	Device handler region extension. Heuristics option. Case shift
	07400-07577	Device handler region extension. 8K High speed close.
Field 1	10000-10377	Block save area/ block save buffer
	10400-10777	Above Screen I/O buffer
	11000-11377	Below Screen I/O buffer
	11400-13377*	Tables, messages, replace/search strings
	*13400-17377	Screen buffer and row table
	17400-17577	Directory Close routines
Field 2	20000-21377*	High speed first/last page and block positioning function. (12k versions only)
	*21400-27577	Offset block array for high speed positioning (12k versions only)

* means approximate address.

On initial load of the VISTA program, the initialization code is located over the screen buffer and the above screen I/O buffer. The VCM device handler region at 6600-7177 contains additional initialization code. The field 2 code is swapped up from low memory on 12k systems.

12. Video Characteristics Modules

12.1 General

The VCM (Video Characteristics Module) program file is a small file occupying three PDP-8 pages in Field 0. The function of this program is to serve as a high level device handler that handles the communications between the main part of the VISTA program and the display terminal. All functions, features and ideosyncracies (timing problems, etc.) are resolved by this VCM file. This means that as new terminals appear, or if more than one terminal is being used at a time, different VCM files can be selected to drive them.

DISC releases a standard set of VCM files in both PAGE8 and PAL8 assembly format. Along with these VCM files is a file labeled VISTA.SD (PAGE8 symbol dictionary) and VISTA.EQ (symbol dictionary modified for PAL8 compatibility). The source code for VISTA is of course in PAGE8 format.

The VCM files are well documented and most users should have little difficulty in building new VCM files, or modifying existing VCM files to handle their terminals.

Scroll Up 17 Blue	Scroll Down 17 Red	Search 35 Gray	Cursor Up 11 ↑
First Page 19 7	Page Up 19 8	Last Page 19 9	Cursor Down 11 ↓
Mark Line 27 4	Pickup 29 5	Putdown 31 6	Cursor Right 11 →
Word Left 15 1	Page Down 19 2	Word Right 15 3	Cursor Left 11 ←
Insert Line 25 0		Home Cursor 13 .	Command Mode 33 ENTER

BACKSPACE	Backspace Delete	21
LINEFEED	Insert Char Toggle	25
DELETE	Delete Character	21
RETURN	Return Code	33
TAB	Tab Right	15
ESC 1	Insert Block	27
ESC 2	Delete Screen	23
ESC 3	Truncate File	33

↑A	Transparent Char	27
↑B	Mark Modified Lines	25
↑C	Abort to OS8	33
↑D	Cursor to Start of Line	13
↑E	Tab Left	15
↑F	Cursor to End of Line	13
↑G	Unmark Line	29
↑H	Backspace Delete	21
↑I	Tab Right	15
↑J	Insert Char Toggle	25
↑K	Delete Word	21
↑L	Delete to End of Word	21
↑M	Return Code	33
↑N	Restore Screen	23
↑P	Invert Case	27
↑R	Continuous Replace	35
↑T	Step Replace	35
↑U	Delete Line	23
↑W	Cursor Away	13
↑Y	Cursor to Next Line	13
↑Z	Close File	31

12.2 VT52 Display Terminals (VT-78 DECstation)

The VCM file for the VT-52 terminal also handles the VT-78 DECstation which is completely compatible with the VT-52. The only difference is that the timing circuit in VISTA for the target columning takes about two seconds rather than the normal 3/4 second.

The VT-52 VCM file has to handle the specific problem of a lack of native insert and delete line functions. The VCM file handles this optimally by examining the position of the cursor when the *Insert* or *Delete Line* functions were issued. The VCM file then issues a scroll up or scroll down command and recopies up to one half the screen (worst case) to delete or insert the line. For example, if there is a *Delete Line* on line 3 of the screen, VISTA will issue a scroll up command at the bottom of the screen. This will cause the top three lines of the screen to be lines 2, 3 and 4 that were originally on the screen. VISTA then recopies the original lines 1 and 2 to the screen, effectively removing the original line 3.

In a parallel manner, if there was an *Insert Line* on line 3, VISTA would issue a scroll down (reverse line feed) on the top line of the screen to force the entire screen to move down one line, leaving a blank line on the top line. VISTA then recopies lines 2 and 3 (which were originally lines 1 and 2) back to their original position and erases line 3. This completes the *Insert Line* function.

Note that the *Insert Block* function is especially useful when a large amount of text has to be inserted on a terminal such as the VT52 that lacks a native insert line function.

Cursor Up 11 ↑	Cursor Down 11 ↓	Cursor Left 11 ←	Cursor Right 11 →

Page Up 19 PF1	Page Down 19 PF2	Scroll Up 17 PF3	Scroll Down 17 PF4
First Page 19 7	Restore 23 8	Last Page 19 9	End of Line 13 -
Mark Line 27 4	Pickup 29 5	Putdown 31 6	Next Line 13 ,
Word Left 15 1	Search 35 2	Word Right 15 3	Command Mode 33 ENTER
Insert Line 25 0		Home Cursor 13 .	

BACKSPACE	Backspace Delete	21
LINEFEED	Insert Char Toggle	25
DELETE	Delete Character	21
RETURN	Return Code	33
TAB	Tab Right	15
ESC O N	Delete Screen	23
ESC O O	Truncate File	33

↑A	Transparent Char	27
↑B	Mark Modified Lines	25
↑C	Abort to OS8	33
↑D	Cursor to Start of Line	13
↑E	Tab Left	15
↑F	Cursor to End of Line	13
↑G	Unmark Line	29
↑H	Backspace Delete	21
↑I	Tab Right	15
↑J	Insert Char Toggle	25
↑K	Delete Word	21
↑L	Delete to End of Word	21
↑M	Return Code	33
↑N	Restore Screen	23
↑P	Invert Case	27
↑R	Continuous Replace	35
↑T	Step Replace	35
↑U	Delete Line	23
↑V	Insert Block	27
↑W	Cursor Away	13
↑Y	Cursor to Next Line	13
↑Z	Close File	31

12.3 DEC VT100 Terminal

The VT-100 VCM file has to handle the specific problem of a lack of native *Insert* and *Delete Line* functions. These can be handled quite simply through the use of the "window" feature when scrolling. By setting the top of the window to the line that we are inserting or deleting, we can issue a scroll up or scroll down command to remove a line or insert a blank line.

The VT-100 terminal should be *SETUP* with auto XON/XOFF control enabled, since VISTA will respond to these codes to prevent the terminal from overrunning. Unfortunately, the VT-100 keyboard appears to lock up after an XOFF and remains locked until an XON is sent. If characters are typed at this point (as might be the case with a fast typist after an insert line from the wordwrap routines) they might be lost. The VCM file for the VT-100 uses the video enhancements of the advanced video option for marking lines (underline for mark, bold for modified lines). If the advanced video option is not present, consult the VCM file which documents the minor changes needed to implement a flag character for modified lines, and either reverse video or underline for marking lines, depending on the cursor setup. [*Note:* The commented changes contain an error. The ZNMV routine must in fact do something: it must send a "0" parameter in order to cancel the underline or reverse video instituted by the ZSPV2 routine. The following code will work properly:

```
ZSPV1,   TAD      ZCHAR
         JMP      DOGRAF
ZCHAR,   "O+40
ZSPV2,   CLA CLL IAC RTL
ZNMV, TAD      ("0
         JMS      ESCBRA
         TAD      ("M+40
         JMP      CHOUT        EJML]
```

The VT-100 terminal can be switched at any time to smooth scroll mode if desired, since the VCM file for the VT-100 is set up to respond to the XON/XOFF sequences. Smooth scrolling on the VT-100 is elegant, but not as fast as the jump scroll mode. Also, in smooth scroll mode, the VT-100 uses the XON/XOFF feature a lot more, which aggravates the keyboard lockup problem. The VCM file sets the VT-100 terminal in ANSI mode and alternate keypad mode at initialization. On exit from VISTA the VCM file will reset the terminal back into VT-52 mode (the normal mode for running a VT-100 under OS/8). See the *Return to OS/8* function for additional notes.

A two instruction patch to the VT-100 VCM file (number of columns and number of lines) can be used to generate an alternate VCM file that will support the 14 X 132 display mode of the VT-100. There is insufficient room in VISTA to support the full 24 X 132 display of the advanced video option. Trial and error can be used to establish the maximum number of lines that the current version of VISTA is capable of supporting (an error message will appear if the VCM file tries to allocate too much room for the screen display). [*Note:* By extensively rearranging the supplied VCM file, it is possible to gain enough space to call internally the ZCLSC routine to clear the screen before implementing the *Return to OS/8* function after a CONTROL/C is received. *EJML*]

Some users of the VT-100 and various look-alike VT-100 terminals have experienced internal timing problems which can cause the VT-100 to get "confused". Even with XON/XOFF control, there appear to be occasions when the VT-100 lets itself get overrun. This can generally be corrected by pressing the *SETUP* key twice followed by a *Restore Screen* function. Check too that the terminal is still in ANSI mode. VISTA switches momentarily to VT-52 mode to issue graphic characters. If the switch back to ANSI mode is lost by the terminal, the ANSI control sequences sent by the VCM file will cause the screen to become corrupted.

The user is free to custom tailor the keypad assignments by simply patching in the appropriate values into the VCM file. The VCM file is distributed with the key assignments shown opposite.

Search 35 Step Replace 35 7	Mark Line 27 Mark Mods 25 8	Pickup 29 Putdown 31 9
Next Line 13 End of Line 13 4	Insert Line 25 Delete Line 23 5	Word Left 15 Word Right 15 6
First Page 19 Last Page 19 1	Page Up 19 Page Down 19 2	Scroll Up 17 Scroll Down 17 3
Program A Program B	Command 33 Restore 23 0	

ESC (or FS)

U	Truncate File	33
V	Delete Screen	23
@	Print Screen	
←	Backspace Delete	21
HOME	Cursor Away	13

↑A	Transparent Mode	27
↑B	Mark Modified Lines	25
↑C	Abort to OS8	33
↑D	Start of Line	13
↑E	Tab Left	15
↑F	End of Line	13
↑G	Unmark Line	29
↑H	Cursor Left	11
↑I	Tab Right	15
↑J	Cursor Down	11
↑K	Cursor Up	11
↑L	Cursor Right	11
↑M	Carriage Return	33
↑N	Insert Char Toggle	25
↑O	Suspend Output	
↑P	Invert Case	27
↑R	Continuous Replace	35
↑T	Step Replace	35
↑U	Delete to Line End	23
↑V	Insert Block	27
↑W	Delete Word	21
↑X	Delete to Word End	21
↑Y	Next Line	13
↑Z	Close File	31

12.4 Ampex Dialog 80 Terminal

The program key functions for this terminal are not fixed permanently: the VCM file programs all 20 (Program A 0-9 and Program B 0-9) at initialization. The program keys send the control character FS (CONTROL/\ one greater than ESC) and one upper case letter (A through T): the VCM file regards FS and ESC as synonymous. Note that the numeric keypad and typewriter keyboard numeric keys are identical on this terminal: the program functions are shown in the illustration with the numeric keypad, but the program keys with the typewriter keyboard numeric keys will perform the same functions. There are five manual ESC (or FS) functions, as shown. If a printer is connected to the printer port, the screen may be printed by pressing the keyboard *Print* key. If the *Print Screen* function is invoked with the ESC @ sequence (this is a VCM file, and not a VISTA, function) the entire screen is printed, then *Page Up and Cursor Away* functions are performed and the new screen is printed: this is repeated until any character is typed or the end of file is reached.