UNIVERSITY OF SHEFFIELD

COMPUTING SERVICES

Using the Sheffield Prime Editor

Wendy Thomson and Peter Mason
Computing Services Department
University of Sheffield
Sheffield
S10 2TN
England

Phone +44 742 768555 extension 4253

Fax +44 742 753899

Telex 547216 UGSHEF G

March 1990

PE1/90

CONTENTS

1. THE SHEFFIELD PRIME EDITOR

1.1 Introduction

The editor used on the Prime computers at Sheffield is not the standard editor supplied by Prime Computers Ltd., but is one developed in Sheffield Computing Services. It consists of two parts, a line editor and a window editor: the line editor is largely compatible with the Prime editor buthas several additional features and is also considerably faster; the window editor is an entirely new facility.
Although this document describes all the facilities available with the window editor, you may also like to try a 'teach yourself' system which is available on all our Prime computers. To use it you should type

```
        LEARN ED
```

and follow the instructions.
This document is reprinted approximately once yearly, so it will happen from time to time that new features have been introduced into the editorsince the latest printing. You can find out about anything new by usingthe built-in Help system: type the line editor command

```
        HELP NEWS
```

This printing of the document reflects the Sheffield Editor at version 8.5.2 in March 1990.
Finally, any comments about the editor would be welcomedby Peter Mason at the address on the title page, or by electronicmail to CS1PM@UK.AC.SHEF.PA.

1.2 General Principles

This document describes how you can use the editor to create new files and also to change existing ones. The Prime Editor can be used in either oftwo ways: when used as a 'line editor' you have to type in directives to describe what you want to do, but when used as a 'window editor' your vdu screen acts like a 'window' on your file so that what you type in acts directlyon your file. (Actually it is not strictly true to talk in terms of acting'directly' on your file, since the editor in fact operates on a temporary copyof yourfile which is only made into a permanent file in your filestore when yourequest it). The window editor is much easier to use than the line editor,and many people never use the line editor. However there are some complicatededits which can be done much more quickly with the line editor, and it isuseful to have a knowledge of its capabilities.
Both the line editor and the window editor may be used either in 'input' mode, to add text to a file, or in 'edit' mode to change the existing contentsof a file. When you are creating a new file you enter the editor in inputmode, but when you are editing an existing file you enter in edit mode.However once you are in the editor you can switch modes, or switch betweenline and window editors, and the distinction between creating a new fileand editing an existing one tends to disappear.
To distinguish the window editor's modes from the line editor's modes we refer to:

input mode to mean line editor input mode. Everything you type in is simply written into the workfile.
edit mode to mean line editor edit mode. Everything you type in is interpreted as editing directives to make alterations to the workfile.
IW to mean window editor input mode. New text is copied intothe workfile, and you may also use the window editing keys to alterwhat you have typed in.
WI to mean window editor edit mode. New text will overwriteexisting text (unless you create space for it) and window editing keysmay be used to make other changes.

Both the window editor and the line editor use a pointer to determine the 'current' point in the file. All new text will be inserted at the current point, and all changes will take effect at the current point. The window editor indicates the current point by means of a cursor (a small white square blob) on your screen. The line editor cannot do this but normally printsout the current line of your file after each directive has been executed.

When you enter the editor, the current point is the beginning of the first line. You can move it about by:

(a) Typing in new text. The current point is always immediately after the last character typed in.
(b) Using the line editor commands to move the pointer.
(c) Use window edit techniques to move the cursor.

It is important to be clear about this concept of current point, so that you do not make mistakes when switching modes. Thus, for example, if youare typing in a new file, and you then enter edit mode to move backwards in the file and make some changes, you must remember to move back to theright place before re-entering input mode to continue with your typing.

1.3 Basic Strategies

As was mentioned in Section 1.2, the only difference between creating a new file and editing an existing one lies in the way in which the editoris called in the first place. Thereafter you may spend some time in input mode (typing in new text) some time in WI mode (altering existing text onthe screen) and some time in (line editor) edit mode typing in directives to alter your text.

1.3.1 Creating a New File

To create a new file, type

```
        ED -W
```

Your vdu screen will be cleared, an empty temporary file will be set up, and as long as you remain in input mode, everything you type in will becopied into the temporary file. You can correct any mistakes on visible text by using the window editing techniques described in Section 2. Ifyou wish to make changes to a part of the file which has disappeared you will have to change to WI or edit mode in order to go back to the offendingline to change

it. (You must then remember to return to the right partof the file before carrying on with your typing).
Alternatively, you can type

```
        SED
```

to enter the editor in screen mode (SED stands for Screen EDit). This has the same effect as typing ED -W with the single exception that it makes you stay permanently in screen-editing mode; the exit to line-editing mode is disallowed. This sounds like bad news, but many new users get very confused if they accidentally drop into the line editor; SED can be very useful on introductory computing courses and for infrequent users. Typing SED is exactly equivalent to typing ED -S.
When you are satisfied with your file you should make it permanent as describedin Section 2.11 or 3.19 depending upon whether you finish up in the windoweditor or the line editor respectively.
For completeness we should perhaps mention that you can simply type

```
        ED
```

to enter the line editor in input mode, and proceed from there. You still type in your file line by line but you cannot correct it so easily as yougo along, and no one uses this method any more.

1.3.2 Editing an Existing File

To edit an existing file, you should type either

```
        ED filename -W

or      ED filename
```

This will cause you to enter the editor in WI or edit mode respectively. Either way your file will be copied into the temporary file with the pointer (cursor) at the beginning of the file. You can now make your changes using the techniques described in Sections 2 and 3, and you can change to input mode if you wish to insert several lines of new text.
When you are satisfied with the file, you can make it permanent, as described in Section 2.11 and 3.19. The editor normally takes a backup copy of any file that you are about to overwrite, so that you can get back to old version if necessary; this process is described in section 2.11.1.

1.3.3 Setting Your Terminal for the Window Editor

Because different terminals deal differently with some of the keys used by the Window Editor, it is necessary for the editor to know what typeof terminal you are using; at Sheffield we ask for your terminal typeat login time and the editor uses this information when deciding whattype it needs to deal with. If you wish to override the chosen terminal typeyou can use a -TERMINAL n optionwith the ED command to set the window editor for your terminal.

e.g.   ED filename -W -T 3

You can also use the line editor command

```
        TERMINAL
```

to find out what value of n you need (see Section 2.3 to see how to useline editor commands from the window editor).

1.4   Using Filenames and Treenames in the editor

Wherever you supply a filename for an editor command, such as when you copy your workfile back into a Primos file, you can give either a filenameor a treename. A filename is just the name of a file in your current directory; a treename is a sequence of directories and a filename whichspecify exactly where the file is located. For example

```
        MYUFD>MYSUBUFD>FILENAME
```

means the file FILENAME in the sub-directory MYSUBUFD of the directoryMYUFD. Some commands, such as the ones for loading and unloading pieces of text, will also accept a buffer number instead of a filename; in that casethe command uses one of the editor's file buffers to temporarily store the text until it is needed again. This is explained in more detail insection 4.
A filename, treename or buffer number may have qualifiers after it to alter the way the text is filed. The possible values are:

Dam   Create the file as a DAM (Direct Access Method) file.
                Sam   Create the file as a SAM (Sequential Access
                Method) file.This is the default for newly created files.
                Append   Add the new text to the end of the file if it
                existsinstead of replacing the file as usual.

The qualifiers are added in parentheses after the name, separated by slashes if necessary. e.g.

```
        FILE filename(append/dam)
```

will copy to the end of the file "filename", creating it as a new DAM fileif it does not already exist.
If you have any 'argument'-class Primos abbreviations the editor will make use of them in pathnames. This means that you can define an abbreviation for all or part of a long pathname that you frequently refer to, and use the short form both in Primos commands and within the editor.
When you are using the window editor you will be asked to type a filename when you save your file to disk; a prompt appears at the top of the screen, possibly giving a default filename that can be used just by pressing the RETURN key.   For this prompt, and also most other window editor prompts,you can decide not to go ahead with the relevant operation after all; todo this you enter the CAN control character by holding down the Controlor CTRL key and pressing X.   The window editor treats CTRL+X as meaning'give up the current operation and wait for next command'.

1.5 Quit Handling

The editor handles "quits" (pressing CTRL-P or BREAK) by interrupting whatit is doing at a safe place and asking for the next command. You can safely break in on the editor at any time. You might, for instance, have typeda command which changes many lines of the file, printing the new version of each; to make it go faster you could break in, type BRIEF to suppressthe printing, and then re-issue the command.

1.6 Entering the Editor

This is a summary of the command you type to enter the Sheffield editor. If you have not used the editor before some of the terms used may beunfamiliar until you have read later sections of this document.
To edit a file or files you type

```
        ED filename_list options
```

"filename_list" is a list of one or more pathnames of files to be modified, separated by spaces.The first one is loaded into the editor's buffer 1, the second into buffer 2and so on. If you give no pathname then the editor starts in 'input' mode inbuffer 1 so that you can type in a new file. The editor prints EDIT or INPUTto show which mode you are in.
"options" available on the command line are as follows (each may be abbreviatedto the '-' followed by the letters in capitals).

-Init pathname	The editor will obey the editor commands contained in "pathname" (in EDIT mode) before asking for commandsfrom the terminal. You can use this to personalisethe editor by including such commands as TABSET to set your preferred tab settings. If you do not give -INIT then the editor will look for a file called PMED_INITin your origin directory and use that as the initfile; if that does not exist (or the -NO_INIT optionis given) then no init file is obeyed.
-No_Init	Disallow the search for the PMED_INIT initialisationfile mentioned above. This option is useful whencalling the editor from a CPL file to avoid anyunexpected effects from commands in the user's PMED_INIT file.
-No_Primos	Disallow the ! and PAUSE commands to enhance the security of the editor if used as part of a turn-keysystem.
-Window	Go straight into the window editor subsystem after obeying any commands in the init file if present.
-No_Line	Do not allow exit from the window editor back to theline editor.
-Screen	Go straight into the window editor and disallow exit(exactly equivalent to giving both -W and -N).
-Terminal n	Set the window editor terminal type to "n", instead ofthe default global variable value or 1.
-Report	The editor will report in a global variable the name used for filing. This option is for

use by systemsthat call the editor.

-Help key	Give help on calling the editor. "key" is optional;if present then that is the first key used in the helpsystem; if omitted then the index page is the firstone displayed.
-TIDY wildcard	Enter the editor directory tidying sub-system.'wildcard' is used to select which files to display;the default is to display all files. If a wildcard is given it must be enclosed in quotes. See section 5 formore information.
-SORT_dtM	The TIDY system normally sorts files alphabetically; this option tells it to sort by 'date/time modified',with the most recent files at the top of the list.
-ReVerse	Tells the TIDY system to reverse the usual order ofsorting.

The -SCREEN option is designed mainly to provide a pure screen editor that is simple to teach to novice users; they do not need to know any line editor commands since they can do everything from within the screen editor. -NOLINE acts like -SCREEN but allows (for example) a CPL macro to obey a few line editor commands before going irrevocably into the screen editor.The special CPL command SED is provided which merely runs the editor withthe option -SCREEN specified in addition to any other options given; SEDis useful for beginners who do not wish to know about line editing.

2. WINDOW EDITOR COMMANDS

2.1 Introduction

In this section we look at the keys you use to perform the various window editing functions. You will find references to key combinationsof two types:

DEL key means you should press the key marked DEL followed by the specified key. Thus

DEL B

means press DEL followed by the letter B.
CTRL+key means you should hold down the key marked CTRL at thesame time as pressing the specified key. Thus

CTRL+N

means hold down the CTRL key whilst you type in your N.

Note that, with the exception of DEL A and DEL a (which are described in Section 2.5) letters may be typed in upper or lower case, e.g. DEL B, CTRL+n etc.
Some operations may only be carried out with the aid of line editing directives.When this is the case you can either use DEL C to execute a single lineeditor directive (see Section 2.2) or you can use the F1 key to transferto the line editor (see Section 2.3). The important point to note is thatline editing directives are al<u>ways terminated by the RETURN key,</u> whereaswindow editing commands are not.
On some terminals there are extra keys which you can use to make window editing easier; for instance, on the Merlinand Televideo terminals you can use the row of keys marked CHAR INS, CHAR DEL etc. for insertingand deleting characters and lines. Those keys are not describedfurther in this document because they may be different for the various types ofterminal, but you can find out about the special keys for your terminalby typing

    Help SUMMary

as a line editor command, or by pressing theF3 function key in the window editor andtyping SUMMARY as a keyword when asked for one. The list given is the onecorresponding to the type of terminal you are currently using.
There may also be some Function Keys on your terminal and again they differ from one terminal to another; you may have been provided with an overlay label to indicate what they do, but Help SUMMary will also tell you.Appendix A of this document includes a list of the actions of the function keys for our Televideo and Merlin terminals.

2.2 General Purpose Commands

The following commands are not used directly in connection with changing the file, but perform various incidental (but essential) operations:

DEL S	Is used to save a copy of your workfile in a permanent file. A message at the top of your screen will ask for the name to beused for the file, and you can either type in the name (followed by RETURN) or use a previously specified name by just pressing RETURN on its own.It is advisable to use this command every 15 or 20 minutes so that a systembreak or similar catastrophe does not result in the loss of hours of editingtime.
DEL C	Allows you to execute one or more line editor directives.The top line of your screen will be cleared and you will be asked to typein your directives. You can type one or more directives, separatingthem by semicolons. Press RETURN at the end to cause the commands to be obeyed,or, if you decide it was a bad idea after all, press CTRL+X to cancel theoperation.
DEL X	Causes the most recent line editor directive to be executedagain.
DEL =	Redraws the current line. Sometimes when the Prime is very busy, the display on the screen acts slightly behind what is actuallyhappening. If you press DEL =, you will be sure which line is the currentline and also what is the current position of the cursor.
F2	This is the F2 function key on the top left-hand side, and redraws the entire screen. This is useful if, for instance, a message fromthe system operator has overwritten the screen.
F3 or DEL H	Enter the editor help system; this allows you to seea summary of the window editing commands, as well as all the other information about the editor. When you type QUIT to leave the help system your original screen display will be restored.

2.3 Changing Modes

The following keys may be used from either IW or WI mode to change to a different mode:

DEL W	switches from IW to WI or vice versa.
F1 or DEL E	leave the window editor and go to the line editor,changing the mode from IW or WI to line editor edit mode.
DEL C	puts you temporarily in line editor edit mode ready for a line editor command. As soon as the command has been executed, you arereturned to your original (IW or WI) mode.

Note that it is not possible to go from IW/WI mode to line editor inputmode: you would have to go to line editor edit mode first, and then switchto input mode.

2.4 Moving the Cursor

"cursor arrows"	The four keys marked with an arroware grouped at the bottom right-hand sideof the keyboard and move the cursor one position in the direction shown. (We cannot print them in this document as there is no corresponding symbol on the printer.) These keys will also repeat if they are held down, but the editor may havedifficulty keeping up if you repeat the "uparrow" or "downarrow" keys and it is better to useDEL V or DEL ^ instead. If the editor does get behind, it redraws the screen to ensure that everything is displayed correctly.

DEL V	Moves the cursor down 12 lines (you can remember thisbecause V points downwards).
DEL ^	Moves the cursor up 12 lines (you can remember thisbecause ^ points upwards).
DEL "uparrow"	Moves the cursor to the beginning of the file.
DEL "downarrow"	Moves the cursor to the end of the file.
DEL "leftarrow"	Moves the cursor to the beginning of the current line.
DEL "rightarrow"	Moves the cursor to one past the last character of thecurrent line so that you can append text.
HOME	Moves the cursor to the top left-hand corner of thescreen.
RETURN	Moves the cursor to the beginning of the next line (but see also Section 2.7).
BACKSPACE	Equivalent to "leftarrow".
TAB	Moves the cursor to the next TAB position. By default these are 6, 12, 30 and 80.
BACK TAB	Moves the cursor back to the previous TAB position.

2.5 Making Changes to the File

There are three ways in which you might wish to change a file:
(a) Typing new text on top of old text
(b) Inserting new text
(c) Deleting old text
Of these, the first is by far the easiest: you simply move the cursor tothe place where you wish to start the changes and then type in your new text; the new will over-write the old. The following keys may also be usedfor changing text:

DEL B	Breaks the current line at the cursor position, to make two lines.
DEL J	Causes the line following the current line to be joined to the current line at the cursor position. Any text to the right of the cursoron the current line will be lost.

DEL A Convert to upper casethe part of the current line which is bounded by the cursor andthe character pointer inclusive, or from the cursor to theend of the line if no pointer is set. You use DEL P (described below) toset the pointer.

DEL a Convert to lower case; the characters affected are defined asfor DEL A. Note that DEL A and DEL a are the only window editing commandswhere upper and lower case are distinguished.

DEL I If you wish to insert new text you can press DEL followed by I to cause new text to be In_s_erted rather than superimposed. Normalovertyping mode is regained by typing DEL I again. Alternativelyyou can create space for new text by using any of the following:

DEL & Inserts one space at the current cursor position.

DEL space Inserts 20 spaces at the current cursor position.

DEL + Inserts a blank line before the line containing the cursor.

DEL " Inserts a copy of the current line.

DEL O Inserts 12 blank lines before the line containing the cursor.

When you have typed in your new text you can delete any spare spaces orblank lines (if you have used DEL space or DEL O) by using:

CTRL+N space Deletes any subsequent spaces.

CTRL+N O Deletes any subsequent blank lines.

To delete unwanted text, you can use the following:

CTRL+N & Deletes one character at the current cursor position.(See also DEL Z below.)

DEL - Deletes the current line.

DEL / Truncates the current line at the cursor position.

You can also delete part of a line by putting a marker at one end of your deletion, moving the cursor to the other end and then deleting what lies between. It does not matter whether the marker is before the cursor orafter it. The keys to do this are:

DEL P Puts a delete marker at the current cursor position.

CTRL+N P Removes the currently active delete marker (not necessarily atthe cursor position).

DEL Z Deletes everything between the cursor and the current marker;the marker will be removed after this action. If you have not set a marker,the single character at the current cursor position will be deleted.

2.5.1 The OOPS Command in the Window Editor

It is possible to undo the effects of recent changes that you have made to the file by using the line-editor OOPS command; when used from thewindow editor this works as follows (see section 3 for normal line-editoruse).
Whenever you start to make changes to a line when using the screen editor a copy of the original version is added to the top of a stack; the stack can hold up to ten copies so usually an old copy will drop off the bottom into oblivion at the same time. You will not normally need to make anyuse of these old versions, but the time will come when you accidentallyoverwrite an important line, or press LINE DELETE when you meant to press LINE INSERT. The OOPS command now comes to your rescue: use DEL C toobey the command OOPS and what the editor does is takes the most recentline copy from the top of the stack and inserts it into the file at the current location. The action is rather similar to the duplicate operation.The old version is inserted instead of overwriting the current line to avoid making matters worse by obliterating anything useful in the new line; withthe old version available you can use the normal screen editing facilitiesto get the file looking how you really want it, which may involve simplydeleting the erroneous changes or combining them together.
You may find that you in fact need an even earlier version of the line, so you can obey the OOPS command again (up to ten times). You can in factobey a command such as

```
        OOPS 3
```

which retrieves and inserts the three most recent copies from the stack. It is then up to you to decide which parts to delete and which to keep.

2.5.2 Search and Replace Operations

If you have many similar changes to make, such as changing the name of a variable throughout a program, then the easiest way is to use the search and replace command REPLACE. This command goes through the file performing the operations:

(1) search for the next occurrence of the search string
(2) ask the user whether to replace it, and do so if requested
It does these repeatedly until the end of the file, or until terminated by theuser. There are options to restrict the operation to certain lines of the file, andto govern the matching operation when searching for the string.The simplest way to use REPLACE is just to obey it with no options (it must beobeyed from within the window editor, so you need to type DEL C REPLACE);you will be prompted for:

(1) The search string; this may include the special characters ! and # thatLOCATE uses
(2) The replacement string.
(3) Options: this is a string of single letters chosen from:

I Identifier type search - only match if found as a word on its own.
C Case independent search - match whether in upper or lower case.

You can just press RETURN if you do not want any options.

When you are doing case-independent replacing, the editor will adjust the caseof the replacement string to match the case of the string found in the file.Thus the command REP/the/these/CI will replace "the" by "these" but it willreplace "The" by "These". The editor will only do this if you type yourreplacement string all in UPPER case or all in lower case, on the assumptionthat you have not explicitly specified what case you want.
The REPLACE operation then proceeds; the cursor will stop on each occurrence of the search string and the message

```
        Replace?
```

will appear; you should then type a single letter:

Y	Yes	replace and continue
N	No	do not replace, but continue
Q	Quit	do not replace, stop searching
C	Continue	do the replace, and continue to search and replace without prompting

When there are no more occurrences of the search string, the command terminates and displays

```
        End of REPLACE
```

You can restrict the REPLACE to a region of the file by:

(1) Mark the first line of the region (DEL M)
(2) Move the cursor to the last line
(3) Perform the replace command
The line marker will be removed when the command completes execution.
When you have used the command a few times you can avoid being prompted by giving the command parameters with the command. For example

```
        REPLACE/search string/new string/CI
```

The format is similar to that for the CHANGE command.

2.6 Moving Text Around the File

If you want to move a portion of text from one part of the file to another, you should do the following:

(a) Put a 'move' marker at one end of the text to be moved.
(b) Move the cursor to the other end of the text.
(c) Put the marked text in an editor buffer or Primos file. If you do notneed the Primos file for any other purpose, it is quicker to use an editorbuffer.
(d) Move the cursor to the place where you want the file to be inserted.
(e) Insert the text.

The keys to achieve this are as follows:

DEL M	Puts an arrow marker on the current line of your file.
DEL U	Copies the text to a Primos file or editor buffer. A prompt will appear at the top of your screen asking you for a filename or buffer number; rememberto press RETURN at the end (or CTRL+X to cancel the operation). The text copied also remains in the file you are editing.
DEL D	Same as DEL U except that the text is deleted from the file youare editing.
DEL L	Inserts the text from an editor buffer or a Primos fileabove the current line of the file.You will be prompted at the top of your screenfor the name of the file or the number of the buffer; remember to press RETURN after typing.

2.7 Tabulating and Indenting

The following keys are used in connection with tabulating and indenting when typing in new text.

TAB, CTRL+I or DEL >	Causes the cursor to move to the next tabulation column.
BACK TAB or DEL <	Causes the cursor to move back to the previoustabulation column.
DEL T	Sets a tab stop at the current cursor position.
CTRL+N T	Clears the tab stop at the current cursor position.

You can also use the line editor TABSET and MODE INDENT commands for tabulatingand indenting:

TAbset n1 n2...	Sets tab stops at the specified columns.The default TAB positions are columns 6, 12, 30 and 80.
MODE INdent	Switches on automatic indenting. This is achieved by changing the normal effect of the RETURN key; instead ofpositioning the cursor at the <u>beginning</u> of the next line, it will be moved along to preserve the indentation of the previous line. Note that this command is only effective when using the window editor and is ignored if used with theline editor.
MODE NOINdent	Switches off automatic indenting.

To use these commands, you should precede them by DEL C as described in Section 2.2

2.8 Simple Text Formatting

The Prime Editor offers simple text formatting facilities which are suitable for text which is to be formatted in a reasonably straightforward way.Thus text is 'tidied up' by the removal of excess spaces between words and by filling lines as much as possible. You can choose the positions of your right and left margins, the format of paragraphs, and the 'mode' (justified or ragged) of the right-hand edge by means of the line editor STYLE directive (described in Section 3.9). Having established your requirements, you can use the following directives and keys to initiate the formatting progress:

DEL M Sets a marker at the beginning or end of the text which is to be formatted.
DEL P Tells the formatter where the current left hand margin is. Youshould move the cursor to the beginning of any line within the sectionto be formatted, and then set this marker by pressing DEL P.This is not necessary if you are using automatic text formatting.
DEL ' Initiates the formatting process. You should move the cursorto the opposite end of the text from the one where you put your M marker,and then use DEL ' to initiate formatting.

You may also use the line editor SYMBOL VISIBLE and SYMBOL VERBATIM commands (see Section 3.9) to change the details of how the formatting is carried out. For further details, see Section 3.9.
There is a line editor command which makes it easier to type in text which is to be processed later:

MODE Margin n Switches the window editor into word processing mode for any new text that you type. The effect is that you can type in your textwithout having to think about pressing RETURN at the end of each line. When aline gets past column n as given with MODE MARGIN (or column 78 by default) the editor splits it automatically at a word boundary so you find yourself typing onthe next line. This allows you to type continuously and your text will befilled into lines of up to 78 columns. You can, of course, still press RETURNto force a new line.
MODE NOMargin Switches off word processing mode.

Word-processing users may also find the STATS and CENTRE commands useful. STATS counts words and lines in a document and CENTRE moves text to the centreof the line; see section 3.9 for more information on these commands.

2.9 Command Repetition

If you wish to execute a window editing command more than once, you can use

```
        DEL R
```

to signify that the next command is to be repeated. You will then be asked for a code to indicate how you wish to terminate your repetition; there are several possibilities, and each one must be followed by RETURN as usual.You can abort the operation at this stage by pressing CTRL+X.The options are:

n where n is a number, meaning repeat n times.
* repeat for ever (i.e. until an error such as beginning or end offile or line is encountered).
M means repeat to marked line (set up by DEL M).
E means repeat until a specified character is found (you will be askedto specify the character).
N means repeat until a character other than the specified one is found(you will be asked to specify the character).

Thus the complete sequence of keys could be

(a) DEL R to initiate the repetition
(b) 200 [RETURN] to indicate that the command is to be repeated200 times.
(c) command the window editing command which is to be repeated.

2.10 <u>Window Editor Macros</u>

You will by now be familiar with the concept of using DEL followed by a character to perform a particular editing operation. Facilities also exist for you to build up your own sequence of commands which are invoked by pressingDEL followed by a chosen character. The commands used in connection withthis are as follows:

DEL (causes subsequent commands to be stored (as well as being executed in the normal way). You will be asked what character is to be used for your macro. You may use any printable character except (or) but if youuse one which is already used by the window editor your macro will over-writethe normal usage until you clear your macro; see "CTRL-N (" or MACCLEAR below. If you use a digit from 1 to 9 you will not have to worry about this sincethe window editor does not use these characters. Remember to press RETURNafter typing in your identifying character.
DEL) terminates the storing of your macro commands.
DEL char is used to execute your macro, where char is the characteryou specified initially.
CTRL+N (is used to delete a macro. You will be asked which macro you wish to delete and should type in the character you chose initially.

In the same way that the window editor has two functions on one key, activated by DEL and CTRL-N, you can define a pair of macros on one key.To define two macros on the "1" key, for instance, you give "1" as the key for the first one and "^1" as the key for the second; the first will beobeyed if you type DEL 1 and the second if you type CTRL-N 1.
The following line editor commands are available for use with macros; they may be used in shortened form as shown by the part in capital letters.You can use themby means of the window editor DEL C command, as described in Section 2.2.

MACPrint abc	displays the sequences of commands which makeup your current macros. You give one space and then a list of macro keysfor the macros you want to be printed.If no keys are given then all macros are printed.
MACClear	clears all your current macros. If any of themhave over-written standard window editor commands, the original usage will be re-instated.
MACSave filename	stores all your current macros in the specified file for future use. If the filename is omitted, the last-mentioned macrofile will be used. The stored file is very similar in format to theoutput of the MACPRINT command.
MACLoad filename	loads (i.e. activates) the macros stored in thespecified file. Any currently defined macros will be over-written by loaded macros of the same name. If the filename is omitted, the last-mentionedmacro file will be assumed. You can use this command in an 'INIT' file(see Section 3.18) to make a set of macros available every time you use the editor.

Window editor macros are stored in a way that does not depend on the type of terminal you are using; you can define and MACSAVE them on one terminal and then MACLOAD and use them on another. If you wish to make some changesto a macro you have defined you can MACSAVE it to a file and then editthe file; you will need to preserve the general layout of the file for it to be successfully MACLOADed again. In saved macro files theleft brace character "{" is stored as double to avoid confusion withcommand keywords which are shown enclosed in braces.

2.11 <u>Leaving the Window Editor</u>

In order to leave the window editor you can use either of the following:

DEL F	Will cause the temporary file to be copied into a permanent file and you will be returned to Primos. You will be asked to type in a name for your file (followed by RETURN) or you can press RETURN by itself ifyou wish to use a previously specified name. Type CTRL+X if you decide notto go ahead with the file operation after all; doing this will leave youin the window editor.
DEL Q	Is used if you wish to abandon your edit, and causes you to bereturned to Primos without the file being changed. You will be asked ifit is all right to lose your edits, just to make sure you have not typedthe command by mistake.

Note that DEL S (as described in Section 2.2) is similar to DEL F in that it is used to save your workfile in a permanent file but, unlike DEL F,it does not take you out of the Editor. It is a good idea to save your workfile every so often in case of machine breaks, lightning strikesor similar catastrophes.

2.11.1 Backup Copies of Files

When the editor is about to overwrite a file it first takes a backup copy so that you can go back to the old one if you need to. The backup copy has the same name as your file with .BAK1 added to the end. If you decide later that your new version was a mistake you can get back to the old by using the COPY command or by deleting the new and using CNAME to change the name of the old. This action of taking backup copies is controlled by the MODE BACKUP command. The usual action is as described above, keeping one backup copy of the file before overwriting. If you type, for example,

```
   MODE BACKUP 3
```

then up to 3 backup copies will be kept, with suffixes .BAK1(the most recent), .BAK2 and .BAK3 (the oldest).You can ask for at most 9 backups to be kept.If you type instead

```
   MODE NOBACKUP
```

then the editor does not take any backup copies.
The action of taking backups proceeds as follows. When you are about to overwrite a file (and have been asked if it is alright to do so if you are using MODE SAFETY), the editor goes into the backup procedure. It will only take a backup of a file you are editing the first time you save it in an editing session, or when you file it at the end if you did not do any intermediate saves; the reason for this is to avoid overwriting the backup with small changes and destroying its usefulness. If you have requested no backups then of course none are taken and the process ends.
The backup procedure first checks whether the oldest backup exists, and deletesit if necessary. It then changes the names of all the backup files to onenumber higher than they were before, and finally copies the file being overwritten to filename.BAK1. When all this has happened it goes ahead and writes the file. If any error occurs in the backup process the user will be asked if they want to go ahead and overwrite the file anyway; if they answer YES then the file will be overwritten without taking a backup.
The editor does not take backup copies when it is being run from a CPL or cominput file, because in general that is not required. After a CPL file reverts to user input with the &TTY directive then taking of backups will occur.
This section has not described absolutely all features of the window editor; if you want to learn about Split Screen editing turn to section 4.

3. LINE EDITOR COMMANDS

3.1 Introduction

In this section we look at the commands you can use in the line editor. These may either be typed in one at a time, each one beingfollowed by the RETURN key, or you may type in several together separatedby semicolons, and pressing RETURN, as usual, at the end:

```
command1;command2;command3
```

Most commands have a permitted shortened form, and these are shown in this document by capital letters.

3.2 General Purpose Commands

The following commands are not used directly for changing the file, but perform various incidental (and essential) operations.

SAVe filename

Saves your edit workfile in the specifiedfile in your filestore. It is a good idea to do this from time to timewhen creating a new file or editing an existing one, since your workfilewould be lost in the event of a system failure, and you would lose allthe work you had done. If the editor already knows the name of your file,either because you are editing an existing file or have already issued a SAVE command or FNAME command (see below) then you can omit the filenamefrom your SAVE command and the current name will be used instead.

EDit filename

copies the specified file into theworkfile ready for editing. If you omit the filename, the workfile willbe cleared and you will be in input mode ready to create a new file. You mu<u>st</u> SAVE your previous file before using this command, although ifyou forget to do so, you will be asked the question

'Is it alright to lose the edits you have done?'

This gives you a chance to say NO, and then SAVE your file. The EDIT commandis mainly used whenyou are editing several files at once. See section 4 for how to do this.

FName filename

Tellsthe editor the name which is to be used for your file if a SAVE or FILEcommand is used without a filename being specified. The FNAME command can be used at any time, and the name will be 'remembered' until required.If you issue an FNAME command with no filename specified, the command will tell you the current name of your file and no further action is taken. A common use for this command is when you are editing an existing file,and wish to give the edited version a new name. You can of course simplytype SAVE or FILE with the new filename specified, but it is all too easy to forget and issue your SAVE or FILE directive with no name specified, and your old file will then be overwritten; it is better to issue your FNAME command right at the beginning of your edit, and then nothing can go wrong.

OOps
This directive is used when you havemade a mistake and wish to undo the effect of the command you have justissued. The current line will be reinstated to how it was before you startedchanging it, so in effect several (consecutive) directives may be nullified.This command can also be used in the window editor where it has a similar purpose achieved slightly differently; see section 2.5.1 for details.

Print n
If n is positive, this directive printsout n lines starting with the current line. If n is negative, the n linesbefore the current line are printed. If n is such that the end or beginningof the file is reached while printing no harm is done:the editor prints .bottom. or .top., as appropriate, and stops printing.This means that if you want to print out the whole file through to the endyou just have to choose a sufficiently large value of n to ensure the endof the file is reached. This directive causes the pointer to move to the point where printing ceases. (See also Section 3.10 for an extended versionof the Print command).

PPrint m n
This directive prints out a range of lines without moving the pointer. The parameters m and n indicate startingand finishing lines respectively, and are specified relative to the current line. Normally the parameters are omitted, and you get five lines before and five lines after the current line. If you want a different range youwill have to specify m and n where a negative number in either positionimplies counting backwards in the file, a positive number implies countingforwards. If only one parameter is specified and it is negative, then printing starts at that line and ends at the current line. If only one parameteris given and it is positive, then printing starts at the current line andends with the specified line. If no parameters are given, printing startsfive lines before the current line and ends five lines after it.

```
e.g.  PP -2 3
```

prints out the file from 2 lines before the current line to 3 lines after it.

Where
Prints where you are in the file.The current line, column and buffer numbers are printed.

Erase character
Kill character
cause the specified characters to be used,instead of backspace and ?, for the ERASE andKILL characters thus causing backspace and ? to be interpreted at face value.KILL is probably the more useful, since you are more likely to want totype in text containing a ? than text containing a backspace.

3.3 Changing Modes

If you wish to change between edit mode and input mode in the Prime editor, you type in a null line; in other words you press RETURN as usual to terminateyour last line, and then press it again without typing anything else in between. If you wish to change to the window editor, you should type

WI to enter the window editor edit mode

IW to enter the window editor input mode

When you are in the window editor (WI or IW mode) you should use F1 or DEL E toreturn to the line editor edit mode, and then press RETURN if youwant the line editor input mode.

3.4 Moving the Pointer

The following commands are used for moving the pointer:

Top
: moves the pointer to the null line at the beginningof the file.

Bottom
: moves the pointer to the last line of the file.

Next n.m
: moves the pointer through n lines and positionsthe pointer at column m. If n>0, the pointer is moved forwards; if n<0, the pointer is moved backwards; if n is omitted, n=1 is assumed. If m isomitted, the first column is assumed.

POint n.m
POint M
: the first form moves the pointer to the mth column ofthe nth line in the file. If m is omitted the first column will be assumed. The second form moves the pointer to the currently markedline (see Section 3.11).

```
Find string            Locate string
Find(n) string         Locate(n) string
NFind string           NLocate string
NFind(n) string        NLocate(n) string
```

These are the main commands used for looking for strings in the file; the'find' commands always look at a particular column of each line (usuallythe first), whereas the 'locate' commands look anywhere along the lines for the string sought. All these commands are followed by a single spaceand then the text of the string sought; if however you have just locateda particular string and want to find the next occurrence then you canomit both the space and the string to save typing.

FIND moves the pointer forwards to the next line which contains the specifiedstring starting in column 1. FIND(n) looks for the string starting in columnn. NFIND looks for the next line which does not contain the stringstarting in column 1. NFIND(n) looks for the next line which does not contain the string starting in column n. The four forms of the LOCATE command are similar, but look for the specified string starting on or after thespecified column.

Thus, for example LOCATE moves the pointer to the next line containing the string starting on or after column 1 (i.e. anywhere), etc. Thespecified string may contain ! characters to mean 'any character' or # characters to mean 'any number of spaces or no spaces'. Thus A!B will bematched with any three character string which starts with A and ends with B,whilst A#B will be matched with any string which starts with A and ends with B and has spaces in between (or the string AB, which contains no spaces). Seealso Section 3.10 for extended forms of these commands.

It may happen when these commands are being executedthat the pointer reaches the top or bottom of the file without finding therequired line (because, for example, the specified string has not been found,or the line number does not exist, etc.); when this occurs, the pointerstops and the message

```
        .top.

or      .bottom.
```

is output, as appropriate. No harm has been done, and you can re-position the pointer as you wish.

3.5 Altering a Line in the File

Append string
: appends the specified string to the end of the current line.

Retype string
: deletes the current line and replaces it with the specified string.

Change/string1/string2/qualifiers
: changes string1 (if it occurs) to string2 in the current line of the file.The action of the command is modified by any qualifiers present after theclosing delimiter, as follows:

G — If G is present then all occurrences of the string in the line arechanged. If G is absent then only the first occurrence is changed.

n — Perform the change on n lines starting at the the current line, instead ofjust the current line.

-n — Perform the change on the previous n lines, starting with the current line.

* — Perform the change on all lines throughout the file.

M — Perform the change from the current line to the current marker (see Section 3.11) inclusive.

I — Perform the change only if the string is found as a word on its own (i.e itis surrounded by punctuation).This is an Identifier change, and allows you, for instance, to change all occurrences of I to J without changing WRITE to WRJTE by accident. Punctuationincludes beginnings and ends of lines, and can be redefined by the PUNCTcommand (see section 3.12).

Examples:

```
CHANGE/x/y/G20  could change alloccurrences of  x  in  the  next  20
                lines to y.
                CHANGE/x/y/*    will change thefirst occurrence of x
                in each line throughout the file to y.
```

CHANGE/I/J/gi* will change all occurrences of I to J throughout the file,but only if I appears as a word on its own and not within another word.

If no qualifiers are specified, only the current line will be changed.The command may be written in the form

```
CHANGE//string/
```

to insert the specified string at the beginningof the line. The string delimiters do not have to be / but may be anycharacter which is not contained in string1 or string2.

Modify/string1/string2/qualifiers

This command is similar to CHANGEin that string1 is replaced by string2 according to the specification of anyqualifiers. In this case, however, the substitution leaves (as far as is possible) the column positions of the remainder of the original line unchanged. Thus if string1 contains n characters, whilst string2 contains m characters,then n>m causes (n-m) spaces to be inserted after string2, and n<m causes(m-n) extra characters in the original line to be overwritten, e.g.

```
10 A$='FIRSTSTRING'
M/FIRST/SECOND/
10 A$='SECONDTRING'
M/SECOND/FIRST/
10 A$='FIRST TRING'
```

Overlay string

superimposes the specified stringon the current line starting in column 1. <u>Note</u>, however, if the specifiedstring contains a space, the original character is left unchanged; a ! must be used to produce a space in the new line. e.g.

```
10 A$='FIRST, SECOND AND THIRD STRING'
O        FOURTH,!FIFTH
10 A$='FOURTH, FIFTH AND THIRD STRING'
```

(note the eight spaces between 'O' and 'FOURTH': The first is the separatorbetween O and its parameter, the remaining seven cause the first seven charactersof the original line to be left unchanged.

EXPand

The EXPAND command replaces any tab characters in the current line by therequisite number of spaces to achieve the same effect. It is useful if you have a file (probably from another site) that contains tabcharacters and you want to convert the file to its correct appearance takingthe tab characters into account. While performing the expansion this command makes use of the tab symbol (set by the SYMBOL command) and the tabulationcolumns set by the TAB command. See section 3.8 for more information.

As an example, suppose you have received a file containing HT characters based on the assumption of tabs every 8 columns; the EXPAND command only affects asingle line, so if you want to convert a whole file you need to loop asfollows:

```
SYMBOL TAB ^211
TAB 9 17 25 33 41 49 57 65 73 81
TOP
LOCATE ^211;EXPAND;*
```

The last line could have been:

```
NEXT;EXPAND;*
```

but it is considerably quicker to avoid expanding any lines not containing tabcharacters.

Gmodify directives

This command offers a way of performing complicated changes to one or more lines of the file; the changes are specified by subdirectives given after the command. Before editing begins, a pointer is positioned at the beginning of the line tobe edited, and an empty line bufferis set up ready to receive the new version. Characters are copied into this buffer as specified by a set of directives, as described below. These directives may (but need not) be separated from eachother by spaces. When all directives have been obeyed, the line buffer is madethe new version of the current line.
The possible directives are as follows:

A/string/ copy the rest of the original line, and then append thespecified string.

Bn move the pointer n spaces to the left; no copying takes place.

Cc copy all the characters up to, but not including the character c.

Dc move the pointer up tothe character immediately before the character c; no copying takes place.

En moves the pointer n charactersto the right; no copying takes place.

F copy the rest of the line.

I/string/ insert the specified string.

J join the next line inthe file to the current line at the current position. Subsequent directiveswill apply to the second line and, in particular, you must remember to useF if you wish to copy the rest of this line.

K break the line at the current position to form two separate lines.

Mn copy the next n characters.

N reverses the next C orD command to mean 'up to the next character which is not character c'.

O/string/ overlays the specifiedstring on the new line. This means that the string is copied to the newline and the pointer moves along the old line the same number of characters.Any spaces in the string will be replaced by the corresponding charactersfrom the old line. If an actual space is required, it should be writtenas a ! character.

R/string/ the same as O/string/ except that spaces are copied as such, and are not replaced by

characters from the old line.
S re-positions the pointer at the beginning of the old line.

Example:
```
original line:  ABCDEFG12345HIJKL
                GMODIFY D1 M5 S C1 E5 F
new version:    12345ABCDEFGHIJKL
                GMODIFY M5 K F
new version:    12345
                ABCDEFGHIJKL
                GMODIFY M5 J F
new version:    12345ABCDEFGHIJKL
```
You must always remember to include an F directive after making a correctionin the middle of a line; if you forget it, you will lose the rest of theline.

3.6 Adding and Deleting Complete Lines

Adding Lines
You can add complete lines of text to a file by moving the pointer to the correct place in the file (remember, information is inserted immediately _after_ the current line), changing to input mode, typing in the extra lines, and changing back to edit mode. This is a bit of a nuisance if onlyone or two lines need to be added, and users may prefer to use the INSERT directive:

Insert string	inserts the specified string as a new line immediatelyafter the current line. If no string is specified, you will go into inputmode and the new lines of input will come immediately after the currentline.
IB string	inserts the specified string as a new line immediatelybefore the current line. If no string is specified, you will go into input mode and the new lines of input will come immediately before the currentline.

After either INSERT or IB has been obeyed, the line that has been inserted becomes the current line.
Information which is stored in a permanent file can be merged with the file being edited by use of the LOAD directive, as described in Section 3.7.
Deleting Lines
One or more complete lines may be deleted from the workfile using either of the following directives:

Delete n	if n is positive the next n lines, including the current line, are deleted. If n is negative, the current line and theprevious n-1 lines are deleted. If n is omitted, n=1 is assumed.
Delete TO string	deletes lines starting with the current line and going forwards until a line containing the specified string isencountered; this line is not deleted.

Note if a mistake is made in specifying thisstring, and no line containing it can be found, deleting will continue rightto the end of the file. Some users prefer to use DUNLOAD (as describedin Section 3.7) to copy deleted information to a file or buffer, so that it can beretrieved, using LOAD, if an accident occurs.

3.7 Moving Text Around the File

In order to copy a portion of your file to a new position in the file, you have to perform the following steps:

(a) Move the pointer to the beginning of the section you want to copy using any of the commands described in Section 3.4 above.
(b) Copy the requisite number of lines to a buffer or file in your filestore.If you wish to retain this part of your file in its original position, youshould use the UNLOAD command, but if you wish to delete it from its originalposition you should use DUNLOAD. Both commands are described below.
(c) Move the pointer to the line of your edit workfile where you wish the text to be moved to(again, use any of the commands described in Section 3.4).
(d) Copy the contents of the buffer or file to your edit workfile at thecurrent pointer position using the LOAD command (see below).

The UNLOAD and DUNLOAD commands are used as follows:

Unload filename n
Unload filename TO string
copies lines of text from your edit workfile to the specified filestore file or editor buffer.If the buffer already contains text which has been modified you will be asked if it is alright to overwrite it. When this command is used inthe first form, n lines of text are copied; when used in the second form,copying terminates when a line containing the specified string is found (this line is not included in the copying process). In both cases copyingstarts with the current line and moves forward through the file; for backwardcopying, see the composite command facility described in Section 3.10 below.
This command does not delete the lines of text from your edit workfile.

DUnload filename n
DUnload filename TO string
this command is similar to UNLOAD, and differs only in that the text which is copied to the filestore file is also deleted from the edit file.

LOAd filename
: copies the contents of the specified Primos file or editor buffer into the edit workfile starting immediately below the current line. The Primos file is not affected; if a buffer is copied then it will be marked as "not modified" because its contents have been copied elsewhere.

These commands do, of course, have uses other than as described above. You can store a frequently-used portion of text in a permanent file and incorporate it in any file by using LOAD. You can use DUNLOAD as a safe method of deleting a large portion of text (if you delete too much by mistake,you can LOAD it back into your edit workfile and start again).

3.8 Tabulating and Indenting

Tabulating and indenting are only used in input mode. The line editor allows you to set up tab stops from edit mode by using the TAB command:

```
        TAbset n1 n2...
```

where n1 n2... are the required column positions for tab stops (the defaults are 6, 12, 30 and 80). When in input mode, you use the \ character to move to the next tab stop position.
Automatic indenting is not available with the line editor, and the MODE INDENT command is therefore described in connection with the window editor (Section 2.7) where it is applicable. It is possible to expand any tab characters stored in a file (by replacing them with an equivalentnumber of spaces); see the EXPAND command in section 3.5.

3.9 Simple Text Formatting

The Sheffield Editor includes facilities for performing simple text formattingon your file. In order to do this you must do the following:

(a) Use the STYLE command (see below) to establish the required formatting.
(b) Move the pointer to either the beginning or the end of the text to be formatted (using any directives described in Section 3.4) and place a marker there (see Section 3.11).
(c) Move to the unmarked end of the text to be formatted.
(d) Use the TEXT command (see below) to initiate the formatting process.

The STYLE Command
The STYLE command takes the form

```
        STyle left right para-indent para-gap mode
```

where

left, right
: specify the left and right margins. The value given for the left margin can be zero, which means that automatic text formatting willbe performed: paragraphs will be reformatted in their current style, subjectonly to the 'right margin' value given.

para-indent
: specifies the indentation on the first line of

a new paragraph.

para-gap	specifies the number of blank lines before a new paragraph.
mode	should be written as FILL or ADJUST (shortened forms For A). The former will fill all lines but will not right-justify them whilst the latter will right-justify them.

Thus for example, if you want your text to be printed between columns 6 and 75 (i.e. leaving a 5-column margin on each side) and you want your paragraphsseparated by one blank line but with no indentation, you would need

```
        STYLE  6  75  0  1  A

or      STYLE  6  75  0  1  F
```

The first would give right-justified text, whilst the second would givea ragged right-hand side.

```
        STYLE  0  75  0  0  A
```

would indicate that you want automatic formatting with the right margin adjusted up to column 75.

<u>The SYMBOL VISIBLE Command</u>

The SYMBOL VISIBLE command takes the form

```
        Symbol Visible character
```

It causes the specified character to be used as a 'visible space' - in other words it appears as a space after your text has been formatted but enables you to specify an exact number of spaces in a particular position (normally the number of spaces between words will be determined by the requirementof right-justification). A common use of the visible space is when you want 'hanging' paragraphs, as shown in the example below.

<u>The SYMBOL VERBATIM Command</u>

The SYMBOL VERBATIM command takes the form

```
        Symbol VErbatim character
```

It causes the specified character to be used to indicate lines that are not to be reformatted. Any lines starting with this character in column 1will be left unchanged during formatting. This facility is useful ifyou have a few lines of text that you have laid out specially in the middleof ordinary text that you want to reformat.

<u>The TEXT Command</u>

The TEXT command is used for initiating the formatting process, and takesthe form

```
        TEXt n
```

where n is the current position of the left-hand margin; this is omitted if you are using automatic formatting because in that case the editor deduces thisinformation from the existing layout of the text. If you are not using automatic formatting you would normally use

```
        TEX 1
```

to initiate your text formatting, since most people type in their text startingat column 1. However, if your first efforts at text formatting are unsatisfactory in some way, but you do succeed in moving the left-hand margin to column 10, say, you could have a second attempt at formatting by using

```
        TEX 10
```

to initiate the process.

<u>The CENTRE command</u>

This command causes the text on the current line to be centred within the margins specified for word-processing; the centreing is achieved by inserting or deleting spaces at the start of the line, the actual text being unaffected. The margins referenced while centreing are those specifiedin the most recent STYLE command, or the defaults if that command has not been used. The CENTRE command can be abbreviated to CE.

<u>The STATS command</u>

The STATS command counts words and lines in part or all of the current buffer.It gives you a printout of the form:

```
    70 lines made up of:
          51 text lines
          19 blank lines
           0 lines beginning with '.'
   466 words in text lines
```

Lines beginning with a dot are ignored when counting words because they are usually word-processor command lines; if you use a differentcharacter for commands then you can use the SYMBOL WPC command to specify it before using STATS.Words are split up in the text by using the setof punctuation characters defined by the PUNCT command; a word is any sequence of characters not in the set of punctuation characters. See the PUNCT command to find out how to change the set of punctuation characters if necessary.
The STATS command normally does its counting in the whole of the file, but youcan restrict it to operate on a region of the file by:

(1) Mark the first line of the region
(2) Move the cursor to the last line
(3) Type STATS

The line marker will be removed when the command completes execution.

<u>An Example of Text Processing</u>

Suppose you want your formatted text to be as follows:

```
(a)   Beginning of first item...
      continuation...
      ....
      ....

(b)   Beginning of second item...
      continuation...
      ....
      ....

(c)   Beginning of third item...
      continuation....
      ....
```

      ....

You can set your left margin to be the column at which the text begins in each line, and you can set a negative indentation of -6 so that the first line of each paragraph begins to the left of succeeding lines, and you can use the visible space to ensure everything lines up correctly.
Thus we could use

```
        STYLE 10 75 -6 1 A

        S VISI @
```

and we would then type our text in the form

```
(a)@@@Beginning of first item...
continuation...
...
(b)@@@Beginning of second item...
continuation...

(c)@@@Beginning of third item...
continuation...
```

The position of the line breaks in each paragraph is immaterial, as theusual filling and adjustment takes place.
The next step is to move to the beginning of your text and set a marker by typing

```
        MARK
```

(as described in Section 3.11). Then move to the end, and type

```
        TEXT 1
```

and off it goes.
Try it and see!
If you have typed in your text with the correct paragraph layout, or youhave already reformatted it as above, then you can use the much simpler automatic formatting to tidy it up after insertions or deletions. Yousimply set the first parameter of STYLE to zero and reprocess the textas above; you will not need to give a left-margin value with TEXT.The editor will go through the text, recognising how you have laid out your paragraphs and refilling lines up to the maximum to tidy them up.Any paragraphs that occupy only a single line are assumed to be already formatted correctly so are not altered.
You can use defaults for the options on STYLE, although you may only omit an option if all succeeding ones are omitted as well - since the editorwill always assume that the first item (if present) is the left margin,the second is the right margin etc. The defaults are:

left margin	column 0
right margin	column 79
para-indent	no indentation
para-gap	one blank line
mode	adjust

3.10 Composite Commands

In this section we look at the use of composite commands which give greater flexibility when printing, searching etc. The command takes the form

```
        oplist searchtype string
or      oplist searchtype filename TO string
```

where oplist is a command made up of asequence of one or more of the following operation codes, whichmust not contain spaces:

P	for print
D	for delete
U	for unload
B	for backwards
C	for case independence
I	for Identifier type locate. If I is given the string will only bematched if it is found as a word on its own in the file (i.e.surrounded by punctuation characters or spaces). This allows you to locate, for instance, "the" without accidentally stopping on"atheist".
N	for not

searchtype is F or L for a FIND or LOCATE type string specificationwith a column number in parentheses if required.

string is the identification string to indicate the termination of the command. The 'filename TO' part is present if the oplist contains a U but not otherwise.

The following shows some examples:

CL Name	will move the pointer forwards until it encounters a line containing the string Name, NAME, name, etc.
CPL Name	as above, but the lines will also be printedas the search proceeds.
ICL Name	the most useful form for locating anidentifier in a program, irrespective of whether you typed it in upper orlower case.
DUBF(7) MYFILE TO SUBROUTINE	will search backwards until a line is reached with SUBROUTINE starting in column 7. All the text up to that line willbe deleted from the edit workfile and stored in the permanent file MYFILE.

3.11 Use of Markers

There are several occasions when you may need to place a marker on your file, such as if you wish to CHANGE your file down to a marked line (see Section 3.5)or if you wish to format your text down to a marked line (see Section 3.9).You may only have one marker in force at any one time, and once set it willremain there until you delete it or use it. The MARK command is used for allmarking operations:

MArk n	marks line number n
MArk	marks the current line
MArk 0	cancels the marker, if any, not necessarily on the current line.

3.12 More General Purpose Commands

In this section we look at general purpose commands which are used less frequently than those described in Section 3.2.

BRief
Verify

Many editor commands cause the current lineto be displayed on your terminal after they have executed,and this can slow things down if you are, for example, using CHANGE to altera lot of lines all at once. The BRIEF command will, however, suppress thisdisplay and will remain in force until the end of the session, or untilVERIFY is used to switch printing on again.

Symbol function character

This is an alternative way of resetting the erase and kill characters, and also permits the setting of other specialcharacters. The characters which may be reset are shown below, with their defaults and meanings:

Kill	?	deletes the current line.
ERase	backspace	deletes previous character.
EScape	^	interprets the following character at its face value.
Wild	!	used with FIND and LOCATE etc to mean 'any character'.
Blanks	#	used with FIND and LOCATE etc to mean 'any number ofspaces or no spaces'.
Tab	\	inserts spaces up to next tab stop.
CPrompt	$	the EDIT prompt when MODE PROMPT is used.
DPrompt	&	the INPUT prompts when MODE PROMPT is used.
Semicolon	;	end-of-line separator when MODE SEMI is used.
COunter	@	the counter symbol when MODE COUNT is used.
Visible	space	used in connection with text formatting to establish

```
                         WPC         .           used  in    connection
                         with the  STATS  command  to  indicate which
                         lines are word  processing  commands  and  not
                         text.
                         VERBATIM    ~           used  in    connection
                         with text  formatting  toindicate lines that
                         are to be left unchanged.

PSymbol
    Can be used to check which charactersare  currently  being  used  as
    special   characters  -  any  symbols  which  are  invisible   (e.g.
    backspace) are written in the form ^nnn  where  nnn  is  their ASCII
    value in octal.
! command
    Obeys the specified Primos command and returns you to the editor.You
    can use !  to obey aCLOSE command, for example, if you find that you
    cannot FILE your edits becausethe file is still open.  You can  also
    use !  to enter the Prime HELP system.Becausethe editor runs as an
    EPF it  is  able  to obey any other Primos command or runany program
    without corruption, but it is possible to get very confused ifyou go
    too deep.
    Primos abbreviations are expanded in the command if you have enabled
    them at Primos level.
Help
    Enters the Editor's HELP system, which contains information about  how
    touse  all  the line editor and window editor commands.  This can be
    very usefulwhen you cannot remember how to use some command, and you
    have forgotten tobring this document to the terminal.   In  general,
    if any  changes  are  madeto the editor the HELP information will be
    kept up-to-date, but of coursethis document cannot be re-printed  so
    easily.
VERSion
    This command prints out the version number of the editor.
ALphabet abc...
    This command is provided mainly for use in  European  countries  where
    theASCII  character  set includes some alphabetic characters outside
    the usualA to Z range;  for instance, in Denmark three  extra  codes
    are assigned forthe special Danish letters.  This command allows you
    to specify the set ofcharacters that is to be affected by the window
    editor commands  which  convertbetween upper and lower case, so that
    the extra characters can be included.
    If the command is given alone then you are asked in turn whether each
    possible alphabetic character is to be included;  you answer Yes or No
    asappropriate.  You can instead give the list  of  characters  after
    the command,separated by one space;  this form is more useful if you
    want to put itin a -INIT file.
    The MODE command with no arguments prints the current alphabetic set.
PUnct ...
    This command allows you to specify the set of characters to be treated
    aspunctuation by commands such as ICL and  c/I/J/gi*  (see  sections
    3.5 and  3.10). The  default  set includes all the usual punctuation
    characters, and will usuallybe alright.  You can either give a  list
    of characters  after  the command,separated from the commandby one
    space, or you can just type the command  and  be  prompted  for each
    possible punctuation  character in turn.  The set always includes the
    space character, and beginning and end  of  line  are  always  treated
```

likepunctuation. If you give the list of characters after the command, you must be careful ifyou need to give any characters which the editor uses as special symbols:e.g. you specify ";" as "^;" and "^" as "^^". The MODE command with no arguments will print the currentset of punctuation characters.

3.13 The Mode Command

The MODE command is used to make changes to the mode of operation of the editor, allowing you to control various aspects of how it operates.Each mode of operation has a name, and the command

```
        MODE name
```

switches on that mode, while the command

```
        MODE Noname
```

switches off that mode. A small number of the modes have extra parameters after the name; these parameters are only meaningful when switching themode on, but the values are preserved if you switch it off and thenon again. Any extra parameters are detailed below.
You can find out the current settings of all the editor's modes by typing

```
        MODE
```

This gives you a detailed display showing which modes are on and which off, together with the current values of any extra parameters where appropriate; you are also shown the current settings of the information associated with such commands as STYLE, TAB and LINESZ, together with a summary ofthe status of the current editing buffer. The display should be usefulif the editor performs in a way that you did not expect, allowing youto check that all the modes have their usual values.
All the possible forms of the MODE command are now given, together with the meanings of any extra parameters.

MODE Backup n
MODE NoBackup
Switches on or off the taking of backup copies of files before overwritingthem with new versions. 'n' is the number of backups to keep for each file overwritten. See section 2.11.1 for a full description of the backup process.

MODE CHain list
MODE NoCHain
Switches on or off the chaining together of a set of buffers so that they aretreated as continuous by commands such as LOCATE. 'list' is a list of buffernumbers to be chained, or ranges of numbers such as 1-20. MODE CHAIN isdescribed in more detail in section 4.3.1

MODE Ckpar
MODE NoCkpar
    The first form checks for parity errors, and is useful if you have (forexample) read a file into the Prime from paper tape. The second form cancelsthe effect.

MODE COlumn
MODE NoCOlumn
    The first form of the command causes a column header display to be printedevery time input mode is entered, or when using PRINT or PPRINT commands; itis helpful when you are editing files which are laid out in particular columns.The second form cancels this effect. The default is MODE NOCOLUMN.

MODE COMo
MODE NoCOMo
    The editor normally switches off the Primos comoutput stream on enteringthe window editor and reinstates it on exit; this is done to avoidfilling the como file with unintelligible control characters which cancause severe problems if accidentally spooled by a novice user. However, there are times when the como output from the window editor is useful (notleast when testing the editor); the editor will leave it switched on ifyou specify MODE COMO. The default is MODE NOCOMO.

MODE COUnt n1 n2 n3 form
MODE NoCOUnt
    This command allows you to use a 'counter symbol' when using the editorcommands APPEND, INSERT, OVERLAY, RETYPE or GMODIFY with A, I, O or Rsub-command. The first time you use the symbol (default is @) it will be replaced by your specified initial value, n (default 1), and each subsequenttime it will be replaced by a value which is incremented by your specifiedincrement, n2 (default 1). The number will be written with n3 digits(default 5). Leading zeros may be printed, suppressed or replacedwith blanks by writing form as PRINT, SUPPRESS or BLANK respectively. Once youhave specified n1, n2, n3 and form, you can use MODE COUNT with no parametersand the current values will be assumed. MODE NOCOUNT makes the counter symbolrevert to being a normal character.

MODE Direct
MODE NoDirect
    The editor normally leaves character echoing switched on when in the window editor because that gives the best response for the user, especiallyif the machine is heavily loaded or a network is involved; it can howevercause problems when the user types ahead while the screen is being updated,although the editor tries to recognise this situation and correct it.If you have a lightly loaded machine with directly connected terminals you may prefer to have the editor perform its own character echoing, givingyou the assurance that the screen is always perfectly up to date and youcan type ahead at any time with impunity. MODE DIRECT switches over to program echoing (for use on direct lines), while MODE NODIRECT switchesback to the default state. This mode has no effect on SSMP terminals because that protocol always gives you the benefit of instantaneousechoing with protection against type-ahead problems.

MODE DISp
MODE NoDISp
When this mode is switched on the editor will display the buffer numberand filename whenever you enter the window editor or switch to adifferent buffer. When switched off, the information is only displayedwhen the editor has chosen a buffer number for you, such as when youuse the NEW command, or 'BUFFER name'.

MODE INdent
MODE NoINdent
Switch on and off the window editor automatic indenting facility.See section 2.7 for more details.

MODE Info
MODE NoInfo
This command is provided to make the editor more similar to the Prime information editor. After MODE INFO, the RETURN key used in line edit modejust moves you onto the next line instead of going into input mode; if you want to go into input mode type the INSERT command. MODE NOINFO cancels theeffect.

MODE Lon
MODE NoLon
The editor normally stops logout notification messages appearing on the screen while you are in the windoweditor, so that the screen is not scrambled; this iswhat most users want because it can be confusing to have these messages overlaid on your screen. However, if you want messages to be allowed though, MODE LON will tell the editor to do so. See also MODE MSG.

MODE Margin
MODE NoMargin
These commands switch on and off the window editor word processing inputmode; see section 2.8 for more information.

MODE MSg
MODE NoMSg
The editor normally stops messages appearing on thescreen while you are in the window editor, so that thescreen is not scrambled; this is what most users wantbecause it can be confusing to have operator messagesoverlaid on your screen. However, if you wantmessages to be allowed though, MODE MSG will tell theeditor to do so. See also MODE LON.

MODE NUmber
MODE NoNUmber
The first form of this command causes the line number to be shown whenever a line is printed;the second form cancels this effect.

MODE Prompt
MODE NoPrompt
The first form of this command will cause the system to output 'prompt' characters at the beginning of each line when it is expecting you to type something.An ampersand (&) is printed for input mode and a dollar ($) for edit mode.The second form of the command cancels this effect. The default is MODENOPROMPT.

MODE Quiet
MODE NoQuiet
The first form of this command switches the window editor into quiet mode; this suppresses window editor messages which are purely informative, buterror messages and prompts are unaffected.You can use this command when you are familiar with the window editor.The second form cancels this effect.
MODE SAfety
MODE NoSAfety
The first form switches on safety checking when the editor writes to files in the Primos filestore: it checks first whether the file exists and if so askswhether you want to overwrite it. The check is not performed if you are justfiling away under the original name of the file, or if the editor is beingrun from a cominput or CPL file (since you cannot generally predict theoccurrence of this prompt). This mode is very useful for avoiding accidentally overwriting important files.
MODE NOSAFETY switches off safety checking; the default is off.
MODE Semi
MODE NoSemi
The first form of this command causes thesemicolon to be used as a newline character in input mode, whilst the second cancels the effect. The default is MODESEMI.

3.14 Editing Files Organised in Columns

There are two commands in the editor which allow you to extract a vertical column of text from a file, edit it separately in a buffer of its own, and then incorporate it back into the original file. You might use these commandswhen editing a column of numbers in a data file, or to produce a multi-column listing.

EXtract buffernumber column1 column2 options
The EXTRACT command allows you to extract a vertical column of text from thefile you are editing and copy it into another buffer so that you can treat itseparately.

buffernumber The number of the buffer into which the extracted text is to becopied.More information about buffers can be found in section 4.
column1 First column number specifying the part of each line to beextracted.
column2 Last column number. On each line this range of columns(inclusive) is extracted and copied to the target buffer.
options If no options are given then the extracted text is not removed from the current buffer. If Blank is giventhen the extracted text is replaced by blank characters. If Delete is given then the extracted text isdeleted from the current buffer; the rest of the line is shifted left to close the space.
examples:
EX 2 41 80 Extract columns 41 to 80 inclusive of each line of the currentbuffer and copy into buffer 2; the current buffer is leftunchanged.

EX 2 41 80 B	As for the previous example, except that columns 41 to 80 arereplaced by spaces after the text has been extracted.
EX 2 41 80 D	The same again, except that columns 41 to 80 are deleted afterthe text has been extracted. Text that was previously aftercolumn 80 is shifted 40 columns left to fill in.

The extract operation normally extracts columns of text from all lines of thefile in the current buffer; if you want to restrict it to just part of thefile then you should mark the line at one end of the part to be affected (usingthe MARK command or DEL M in the window editor), move to the line at the otherend and then obey the extract command. This will restrict the extractoperation to the lines from the current line to the marked line inclusive.

MErge buffernumber column1 column2 options

The MERGE command allows you to merge two file buffers side-by-side. It takesthe lines from a specified buffer and overlays them one at a time onto successive lines of the current buffer, starting at the current line, replacingthat part of the text of each line within the given range of columns.

buffernumber	The number of the buffer containing the text to be merged into the current buffer. More information about buffers can be found in section 4.
column1	First column number specifying the part of each line to bereplaced.
column2	Last column number. On each line this range of columns(inclusive) is replaced by the text from the file being merged.
options	If no options are given then the merged text overwrites the text in the current buffer. If Insert is given thenthe merged text is inserted into the lines of the currentbuffer; the rest of the line is shifted right to make space.

examples:

ME 2 41 80	Merge the contents of buffer 2 onto this buffer, replacingcolumns 41 to 80 of each line. Each line of buffer 2 is overlaid onto the corresponding line of the current buffer,starting at the current line,replacing columns 41 to 80 with the new text from buffer 2.
ME 2 41 80 I	As for the previous example, except that the new text isinserted into columns 41 to 80 inclusive; the text that waspreviously after column 40 is shifted 40 columns right to makespace.

The merge operation finishes when all the lines of the specified buffer have been used; if the bottom of the current buffer is reached during the mergethen the buffer is extended with blank lines to allow the merge to continue.

3.15 Command Repetition

If you wish to execute a sequence of commands several times, you can type them in all together in the form:

```
        comm1;comm2;...commn;*m
```

where comm1,comm2,...commn are the commands to be repeated, and m is the number of times they are to be executed. You can omit the m if you want the commands to be repeated until the end of the file is reached

```
e.g.    L NAME1;C/NAME1/NAME2/;*
```

will go through your entire file, locating occurrences of NAME1 and changing them to NAME2. Be careful to avoid doing the following:

```
        C/NAME1/NAME2/;*
```

as this will go on for ever (unless you break in). You can 'nest' repetition,by including the * or *m as many times as you like in a sequence. Thus,for example

```
        C/0/ /;*3;N;*
```

will change the first three occurrences of 0 to a space throughout your file.
Another way of achieving repetitive editing is to use string buffers. You can put a complicated sequence of editing directives into a stringbuffer and thenexecute them by quoting the string buffer name. The commands for doing this are:

MOVe sbuffer /string/   Copies the specified string (usuallyyour sequence of editing commands) into a string buffer.You can use as delimiter any character that is not in your string; / isusually used.
MOVe sbuffer1 sbuffer2 Copies the contents of string buffer 2 intostring buffer 1.
Xeq sbuffer   Causes the editing commands in the specifiedstring buffer to be executed. If no string buffer is specified, the mostrecent line of edit commands is executed again.
Print sbuffer   Prints out the contents of the specified string buffer.
Print All   Prints all string buffers that are not empty.

You can use up to ten string buffers called STRA, STRB, ... STRJ or STR.1, STR.2 ... STR.10. In addition there are two special stringbuffers, namely EDLINwhich always contains the last command (or sequence of commands) typed in,whilst INLIN contains the current line of your workfile. Some people do clever things like putting a sequence of editing directives into a file, LOADing this to their workfile, so that they are contained in INLIN, andthen issuing an XEQ INLIN command to execute them. Remember to delete the line from your workfile if you do this! The most common use of these commandsis to use Xeq on its own to repeat the most recent edit command,or to copy a sequence of commands to a string buffer and execute that repeatedly.

```
e.g.    MOVE STRA /N;L NAME1;C!NAME1!NAME2!/
        XEQ STRA
```

3.16 Line Editor Macros

The editor provides by means of the OBEY command a simple means of substitutingvalues of variables into commands so that useful effects can be achieved. TheOBEY command works in conjunction with standard variables so you can use such things as 'current line number' in your editing; it is intended mainly for usein window editor macros. To use a standard variable in a command the commandmust be issued by OBEY in the form

```
   OBEY 'command1; command2...'
```

You can use any punctuation character to enclose the line of commands; we usedquotes in the example.
To illustrate the use of OBEY we will use the standard variable LINENO which contains the current line number in the file.To insert the line number at the start of the line type

```
   OBEY 'C//%lineno% /'
```

The same effect could be achieved by

```
   MOVE STRA 'C//%lineno% /'
   OBEY STRA
```

To see how this works, suppose we are on line 23 of the file and we type the above command. First of all the OBEY command copies

```
   C//%lineno% /
```

into a temporary string buffer. It then goes through the buffer looking for variables enclosed in % characters. In this case it recognises the variable LINENO (it can be in upper or lower case), and replaces it by its value of 23.

```
   C//23 /
```

This command is then obeyed, inserting '23 ' at the start of the line.
To extend this example, if you wanted to insert line numbers throughout a fileyou could type

```
   POINT 1
   OBEY 'C//%lineno% /';NEXT;*
```

Note that the very similar command

```
   OBEY 'C//%lineno% /;NEXT;*'
```

would not have the same effect; it would in fact insert '1 ' before every line. The reason is that variable substitution occurs when the OBEY command isexecuted, which is once for each line in the first case, but only once altogether in the second.
Further examples:

OB 'POINT .%indent%'	Move to the first non-blank character on the current line.
OB 'NF(%colno%) '	Find the next line that is non-blank in the current column. This can be used for finding an END in a block-structuredlanguage.
OB 'BNF(%colno%) '	Ditto, but move up the file.
OB 'BUFF 6;SAVE;BUFF %buffno%'	Switch to buffer 6, save it, and switch back to the current buffer.

3.16.1 <u>Editor Standard Variables</u>

The following variables are available for use in the OBEY command. When used they are enclosed in % symbols. If you want an explicit % symbol in the command then use %%. The variable names are recognised in upper or lower case.

BUFFNO	The number of the current buffer.
LINENO	The line number of the current line.
COLNO	The column number within the current line.
MLINENO	The line number of the marked line, or zero if no line is marked.
PCOLNO	The column number of the character marker, or zero if none.
PLINENO	The line number of the character marker, or zero if none.
INDENT	The indentation level of the current line; this is the column number of the first non-blank character on the line, or one if theline is all blank.
NEXTNO	The next number extracted from the current line of the currentbuffer. The editor starts at the current position on the line andworks forwards until a decimal number (positive or negative) is found; it gives the value of the number or zero if none is found.This is useful for picking up line numbers from listing files.
FILENAME	The filename of the file being edited in the current buffer,exactly as specified at start up or on the EDIT or NEW command.
INLIN	The complete text of the current line of the file.
EDLIN	The complete text of the last line of commands obeyed.
STRA...	The contents of a string buffer. Can be specified as STRA to STRJor STR.1 to STR.10.

You can also access standard Primos global variables; these are recognised bythe fact that the name begins with a dot. For this to work you must have activated a global variables file by using the DEFINE_GVAR command.
For example, outside the editor you obey

```
   DEFINE_GVAR globals
   SET_VAR .myglobal := myvalue
```

In the editor:

```
   OBEY 'I %.myglobal%'
```

will insert a line 'myvalue' into the file.

3.17 The Spelling Checker

The SPELL command, which may be abbreviated to SPE, goes through your file checking that the words you have typed are in the dictionary. You can use it to check for spelling and typing mistakes in your documents. It goes through the following stages:

1: It reads the dictionary of around 80000 words into memory so that it can beaccessed rapidly; the dictionary stays in memory, so any further uses ofSPELL in the editing session will not need to read it again. If there isnot enough memory for the dictionary you will get an error message; youwill need to free some dynamic segments using the REMOVE_EPF command, orleave the editor and obey ICE to clear things out.
2: It reads any list of glossary words that it can find, so that it does notkeep on reporting the same words over and over again when you already knowabout them. See below for more about user dictionaries and glossaries.
3: It scans through the text, splitting it up into words and checking them inthe dictionary. It normally starts from line one and scans the whole file,but you can restrict it to part of the file as described below. It keeps arecord of all different words so that it only has to look each one up once.
4: When it finds a word that is not in the dictionary it asks you what to doabout it. You can correct it or tell SPELL that it is alright as it is.
5: At the end of the document or section indicated it reports how many wordswere checked, and updates the user dictionary or glossary if you have addedany words.

The SPELL command is very good at finding spelling and typing mistakes in a document, but it does not remove the need to check documents thoroughly: it cannot, of course, tell you whether you have used "their" when you should haveused "there". Its great advantage is the speed with which it draws yourattention to the suspect words in the document. SPELL automatically skips over any one-letter words, and also any 'words' containing non-alphabetic characterssuch as numerals. Lines beginning with '.' are assumed to be word-processing commands and are ignored; the '.' can be changed to something else by theSYMBOL WPC command.
SPELL can be run from the line editor or by using DEL C from the screen editor. The action is almost identical, the single difference being when you type a response to indicate what to do with a non-dictionary word: in the

lineeditor you type a one-letter command followed by pressing RETURN; in thescreen editor the RETURN is not needed. The command letters you can type are:

A Add the word is alright; add it to the glossary or user dictionary(see below)
S Skip the word is alright; skip over it and continue checking
R Retype the word is wrong; retype it. You will be prompted for the correct spelling of the word, and checking will continue fromthe start of the replacement word so that the new spelling ischecked.
Q Quit terminate the spell-checking session.

You can restrict the SPELL command to a region of the file by:

1: Mark the first line of the region
2: Move the cursor to the last line
3: Perform the SPELL command
The line marker will be removed when the command completes execution.

3.17.1 User Dictionaries and Glossaries
Documents almost always contain words that are not in the dictionary; they might be proper names or technical terms, or just relatively uncommon words. It would be very tedious to keep on telling SPELL about them, so there are waysof giving SPELL lists of words that are alright even though they are not in the dictionary. You can build a glossary for an individual file that contains lotsof special words, or you can build user dictionaries either for all files orjust all the files in a particular directory. When SPELL starts it looks for adictionary or glossary in the following order:

1: If you are editing a named file SPELL looks for a file named ==.GLOSSARY inthe same directory. For example, if your file is called MYDOC.RUNI then itlooks for MYDOC.GLOSSARY, and if it is called TEXT it looks for TEXT.GLOSSARY, both in the same directory as the file. If you want to use an individual glossary file then just create it as an empty file before using SPELL.
2: It looks for a file called EDITOR_USER_DICTIONARY in the directory that youare attached to; this would contain glossary words for all files in thedirectory. If you want a user dictionary for a particular directory, justcreate it as an empty file before using SPELL.
3: It looks for a file called EDITOR_USER_DICTIONARY in your origin directory.This would contain glossary words for all files everywhere. The editor willcreate this file for you automatically if it does not exist.

If none of these is found then you will start with an empty list of glossary words. During the spelling session you will probably come across words that are reported as not in the dictionary but you know are in fact alright. If a word is unlikely to be encountered again then it is best to Skip it, but otherwise Add it to the glossary so that any future occurrences will not be reported.
At the end of a session in which you have added words to the glossary, the SPELL command will update whichever file it read the words from (after asking first whether it is alright to do so). If it did notfind any glossary file

or user dictionary, then it will create one for you inyour origin directory. These files are merely lists of words so you can editthem or delete them if you wish. The dictionary itself is kept in a highlycompressed form which cannot be edited.

3.18 Initialising the Editor

If there are any Line Editor commands which you frequently find yourself typing in before starting your edit (e.g. MODE INDENT, TABSET etc.) youcan arrange for them to be automatically executed for you at the start ofyour edit. To do this, you should put them all into a file called PMED_INIT in your origin directory (i.e. the directory that Primos attaches you towhen you first login). From now on, whenever you enter the editor eitherdirectly or as part of some other system (such as a mailer or command environment), the commands in the file will be obeyed before startingto customise the editor to your requirements.
If at some time you need to specify a different initialisation filefor some special purpose you can add the option

```
        -INIT pathname
```

to your ED command. (Pathname is the Primos pathname of the file containing the editing directives.) You can use this facility regardless of how you are entering the editor:

```
e.g.    ED textfile -INIT pathname    for line editor edit mode.

        ED -W -INIT pathname          for window editor input mode
```

The editor always goes temporarily into line edit mode while itexecutes the commands in your initialisation file, and then transfer you to the appropriate mode as indicated by therest of your ED command.
Systems programmers should note that the commands a user includes in the PMED_INIT file can cause trouble when running the editor from a CPL procedure; for instance, MODE SAFETY could be switched on causing an unexpected prompt to appear. You can solve this problem by using theoption -no_init to suppress the search for the PMED_INIT file, or alternatively specify a special init file using the -init option.

3.19 Leaving the Line Editor

In order to leave the line Editor you should use one of the following:

FILe filename — causes the workfile to be copied into a filestore file with the specified name. You can omit the filename if you wish touse a name you have specified previously (on a SAVE or FNAME command as described in Section 3.2) or if you are editing an existing file and wishto keep the same name.
Quit causes the edit to be abandoned without the workfilebeing made permanent; you can do this when things have gone badly wrong and you decideit would be easier to start again. If you have made some changes to your file, you will be asked if it is

Alright to lose the edits you have done?

as a safeguard against typing QUIT by mistake.If you were using the editor just to inspect a file with no intention ofchanging it, then of course you finish by typing QUIT.

QF is the same as QUIT except that the edit isabandoned without question. Do not get into the habit of using this command unless you are sure you will not use it by mistake!

The editor keeps backup copies of any files that you overwrite while editing, so that you can go back to the old version if you make a mistake. See section 2.11 for more information on this topic.

3.20 An Example of Line-Editing

The following shows a simple example to illustrate various editing techniques.The lines typed in by the user are shown by underlining.

```
OK, ed edexample
[Sheffield Editor version 8.5.2]
Copyright (c) University of Sheffield 1990
Edit
print 99
.top.
The Microbe is so very small
You cannot make him out at all,
But many sanguine people hope
Th see him through a microscope.
His jointed tongue that lies beneath
A hundred curious rows of teeth:
His seven tufted tials with lots
Of lovely pink and purple spots:
Of lovely pink and purple spots:
On each of which a pattern stands,
His eyebrows of a tender green:
All these have never yet been seen-
But scientists,
Assure us that they must be so....
Oh1  Let us never never doubt
What nobody is sure about!
.bottom.
top                                   return to top of file
locate see                            locate line containing 'see'
Th see him through a microscope.
change/h/o/                           change h to o
To see him through a microscope.
next 4                                move forward 4 lines
Of lovely pink and purple spots:
delete                                delete current line
find But                              find the line beginning with 'But'
But scientists,
append who ought to know,             append ' who ought to know'
But scientists, who ought to know,
next 2                                move forward 2 lines
```

```
Oh1  Let us never never doubt
change/1/!/                          change 1 to !
Oh!  Let us never never doubt
print -99                            print the file to check it again
.top.
The Microbe is so very small
You cannot make him out at all,
But many sanguine people hope
To see him through a microscope
His jointed tongue that lies beneath
A hundred curious rows of teeth:
His seven tufted tials with lots
Of lovely pink and purple spots:
On each of which a pattern stands,
His eyebrows of a tender green:
All these have never yet been seen-
But scientists, who ought to know,
Assure us that they must be so....
Oh!  Let us never never doubt
bl tials;c/ia/ai                     locate backwards the line containing
His seven tufted tials with lots     'tials' and change ia to ai
His seven tufted tails with lots
f On; I Composed of forty separate bands: find the line starting with 'On' and

On each of which a pattern stands,       insert the new line 'Composed of'
bl seven                              locate backwards the line containing
His seven tufted tails with lots      'seven'
p6                                    print the next 6 lines to check again
His seven tufted tails with lots
Of lovely pink and purple spots:
On each of which a pattern stands,
Composed of forty separate bands:
His eyebrows of a tender green:
All these have never yet been seen -
fil
EDEXAMPLE
```

There is, of course, no unique way of editing a file, and the above shows only one way to correct the typing mistakes made. Thus, after printingout the file ('99' was a convenient number to choose for the number of lines to be printed - there is no need to count up to find the correct number)we then returned to the top of the file, and worked through it correctingvarious errors. In order to print it out again, we used 'print -99' sothat counting went backwards instead of forwards and we could look at thepart of the file we had changed. This left us back near the bottom of thefile again, and we had to use the composite backwards locate to reach the line containing the erroneous 'tials'. After making one or two more corrections,we again use backwards locate to retrace our steps and then print out the six lines containing the recent changes. We could, of course, have usedthe NEXT command with a negative number to move backwards in the file.
You will notice, perhaps, that all the editing commands are typed in lower case; we could have used upper or lower case but lower is more convenient when editing a lower case text file such as ours, since the insertions, changes etc. will normally be written in lower case.

4. EDITING SEVERAL FILES AT ONCE

4.1 General Principles

The Sheffield Editor allows you to keep many files in memory at once, each one being kept in a separate 'buffer'. You can put the required filesin buffers by means of an extended form of the ED command:

```
        ED file1 file2...
```

where the first file is placed in buffer 1, the second in buffer 2 etc. You can also put additional files into buffers when you are already inthe Editor, as we shall see shortly.
Instead of giving just a list of files you can use the Primos features of wildcarding, treewalking and iteration to specify many files at once. These features are described in the Primos Commands reference guide. For example,

```
        ED @@.PAS
```

will load all files with the suffix .PAS in the current directory, and

```
        ED *>@@>@@.(PAS F77)
```

will load all files with suffixes .PAS or .F77 in all directories below the current. If you use any of these features on the command line the editor reports how many files have actually been loaded into buffers for you; this is useful information if you intend to use MODE CHAIN.The NEW command can also make use of these facilities for loading multiplefiles.
When you first enter the Editor, your current buffer is buffer 1, and all editing commands you type in will be applied to that file. You can change the current buffer by typing

```
        BUFFer n
```

where n is either an integer specifying the buffer number or a string of characters matching all or part of the pathname of the file in the required buffer; matching treats upper and lower case characters as identical.If you have already placed afile in the specified buffer, that file will be edited by your subsequent editing commands. If you have not placed a file in the buffer, the bufferwill be empty; you can go into input mode and create a new file in it, oryou can copy an existing file into it by typing the editor command

```
        EDit filename
```

You will quite commonly want to perform the two operations of first finding an empty buffer and then editing a new file in that buffer; you can do this with the one command

```
        NEW filename
```

The NEW command looks for the first empty or unused buffer and switches to it, and then loads the given file into that buffer. It is equivalent toa BUFFER command followed by an EDIT command. When you give a filename or treename for the NEW command you can use Primos features such as wildcarding and iteration to make it load in many files instead of just one. For obvious reasons this does not apply to the EDIT command.

RELoad

The RELOAD command loads a fresh copy of the file from disc into the current buffer, after checking whether it is alright to lose any editsyou have already done. RELOAD is useful if you have started makingsome changes to a file but you decide that what you have done is wrongand you would be best starting afresh. Typing RELOAD is exactly thesame as typing 'ED filename', where filename is the name of the fileyou are working on.

You can find out which buffers are currently occupied by issuing a BUFFER command with no buffer number specified. This will also indicate the current buffer by means of an arrow (->) and will indicate which buffers have been modified (see Section 4.4) and which files are still open for reading.If you are using split-screen editing this command also indicateswhich buffers are displayed in which windows on the screen.The file copied into a buffer stays open until you first go to the bottomof the file; this is done to save having to read the whole file in before youcan start editing, as you may not want to look all the way down a huge file.

If you have loaded many files into the editor and you want to browse round looking at them in turn, you may find these commands useful: the NEXTBUFF command switches you to the next buffer that contains a file, cycling back to buffer 1 at the end of the list. The PREVBUFF command cycles round in the opposite direction. Abbreviations are NEXTB and PRE.

4.2 Making Your Workfiles Permanent

You can issue SAVE or FILE commands at any time to make a permanent copy of the file which is in the current buffer. FILE will of course cause youto leave the Editor, so SAVE is more likely to be useful, even if you have finished editing the file in the current buffer (since you will probablywant to move on to a different buffer). You do not have to SAVE or FILEa file before switching to a different buffer, since yourfiles are kept intact in their buffers. However, you are recommended touse the command

```
        SAVEAll
```

from time to time, as this will SAVE all files which have been tagged 'modified'(see Section 4.4).When you finish editing you can use the command

```
        FILEAll
```

which is equivalent to a SAVEALL followed by a QUIT, but the QUIT is only triedif the SAVEALL worked successfully.

SAVEALL and FILEALL use the current names specifiedfor your buffers, and so they will fail to save a modified buffer if youhave not yet given it a name. In that case you will need to switch to thatbuffer and use FNAME or SAVE with a filename specified (see section 3.2).

Sometimes when you are editing many files simultaneously you need to break off for a while, and it can be quite a complicated job gettingthe edit restarted exactly where it was when you finished. There are two editor commands which help you here.

```
        SAVEState filename
        LOADState filename
```

SAVESTATE first performs a SAVEALL operation to save all modified buffers to files, and then writes a file of editor commands thatcan be used later to restore your current position in the editor. TheLOADSTATE command is used to read the file of commands. After saving andrestoring in this way you will have all the same files loaded into the samebuffers and positioned exactly as before.
The SAVEALL operation will fail if you have any buffers containing modified text but not assigned to filenames. When writing the file of commands, you will be warned if any buffers contain text but do not have filenames, since itwill not be possible to restore these. They may just be some text dumped fromthe file you are editing and not needed again, in which case you can ignore the warning, but if you do want them again you should specifically save them tofiles and perform the SAVESTATE command again.
The LOADSTATE reads commands from the file written by SAVESTATE; you should enter the editor with no file being edited, make sure that you are in edit mode, and obey the LOADSTATE command. It may also be obeyed from within the window editor, in which case each line of commands is obeyed as thoughit were preceded by DEL C or the OBEY COMMAND function key.
The first time you use either SAVESTATE or LOADSTATE in an editing session you must give the name of a file, but you can omit it laterif you wish to use the same file again.
The LOADSTATE command merely reads lines of commands and obeys them, which can be useful in other situations. For this purpose the COMINPUTcommand is also provided, which acts in exactly the same way with the single exception that it does not use the default filename. COMINPUTcan read lines of commands from a file or an editor buffer, so itcan be used to set up and execute complicated edits.

4.3 <u>More Commands for Use with Buffers</u>

Other commands used in connection with buffers are as follows:

LOAd n — is similar to LOAD, as described in Section 3.7, but this form is used to copy the contents of buffer n into the current buffer. Buffer n is tagged 'unmodified' (even if it had been modified sincethe last SAVE or SAVEALL) since its contents have been copied elsewhere (the current buffer is of course tagged 'modified').
Unload n m — is similar to UNLOAD, as described in Section 3.7, but this form is used to copy m lines from the current buffer into buffer number n.If buffer n contains modified text you will be asked if it is OK toproceed.
DELBuf n1 n2... — deletes the specified buffers. You can give a list of buffer numbers or specify ranges such as 3-10, meaning all buffers from 3to 10 inclusive. If any buffer to be deleted contains modified text you willbe asked if it is alright to lose the edits. If you delete your currentbuffer you will be switched to the next buffer containing text.
SIZE — list all buffers that are in use, indicating thesizes of the workfiles stored in them. The sizes are given in lines, in Prime words

(2 characters per word), and in records of 1024 words. Thetotal size of all buffers is also given. If any of the files have not yet been completely read in there will be a short delay before the information appears.

The window editor commands for moving lines of text may also use a buffer as intermediary; you give the buffer number instead of a filename.See section 2.6 for details of the commands.

4.3.1 Buffer Chaining

When you have several files in editor buffers you sometimes find that you wantto make the same or similar changes to them all. This can prove tediousbecause of all the buffer switching and retyping of commands necessary. MODECHAIN can make it a lot easier.
MODE CHAIN has the effect of making a set of buffers appear like one long fileas far as certain commands are concerned. You can specify that, for instance,buffers 1 to 10 inclusive are to be treated as a chain; if you then type TOPthe current position moves to the top of buffer 1, not just the top of the current file, and of course BOTTOM moves you to the bottom of buffer 10. ALOCATE command will move on to the next buffer in the chain if it hits thebottom of the current buffer, and so on.
A typical use of buffer chaining would be to change the name of a variable throughout a large program kept in many source files, or to do systematic changes to the whole of a document when each chapter is kept in a separate file.
To switch on buffer chaining you type, for instance,

```
   MODE CHAIN 1-10
```

which indicates that any files in buffers 1 to 10 are to be treated as a chain.It does not matter if some of the buffers are empty as buffer chaining skipsover any empties when looking for the next buffer. The list of buffers can bemore complicated, such as

```
   MODE CHAIN 1 3 7 10-12 2 4
```

which chains the buffers 1, 3, 7, 10, 11, 12, 2 and 4 in that order. To switchoff chaining type

```
   MODE NOCHAIN
```

If you have already used chaining in this editing session you can switch it back on again with the same buffers in the chain by typing

```
   MODE CHAIN
```

but of course this would be an error if it was the first time you had used the command.
When buffer chaining is in operation the action of the following commands is affected.

TOP Moves to the top of the first buffer in the chain.
BOTTOM Moves to the bottom of the last buffer in the chain.
LOCATE (and FIND and all variants) Move onto the next buffer in the chain when they hit the bottom of the current buffer, orvice-versa if moving backwards.
CHANGE (and MODIFY) If you give the '*' qualifier after the command itwill apply to all buffers in the chain instead of just the wholeof the current buffer.
REPLACE If no line is marked the replace operation will apply to the whole chain of buffers instead of just the current one.

If buffer chaining is in operation but the current buffer is not part of the chain then all the above commands apply as usual to just the current buffer. All commands not mentioned above, and in particular all window-editor commands,apply to just the current buffer.
Whenever the buffer chaining operation switches to a different buffer it announces the new buffer number and filename so that you can keep track of where you are.

4.4 Split Screen Editing

The editor allows you to display and modify the contents of more than one editing buffer at the same time; this is called 'split screen editing' becausethe screen is split into two or more 'windows', each one displaying thecontents of an editing buffer. Split screen editing can be very useful whenamending several files which are similar or related in some way, because youcan keep all the files visible at the same time. Of course there are limits to how many windows can be used, dictated by the minimum window size that isviable.
Split screen editing imposes a slight penalty on response time which is most noticeable on network terminals. The reason is that the editor has to performits own echoing to keep the split screens up to date. There will be nodifference however if you normally use MODE DIRECT or if you use an SSMP terminal. In fact the SSMP protocol provides special facilities for split screen editing which are utilised by the editor, mainly when the split is horizontal.
To enter split screen editing you use the SPLIT command, which may be obeyed either before entering the window sub-system, or via DEL C from within the window editor. A typical SPLIT command is:

```
SPLIT 1 2 VERTICAL
```

which tells the editor to split the screen vertically into two windows, one containing buffer 1 and the other buffer 2. Buffer 1 will occupy the left halfof the screen and buffer 2 the right half. You can give a list of up to eightbuffers to be displayed on the screen; more than eight implies a window sizetoo small to be useful.
The word at the end of the command indicates whether you want a vertical or a horizontal split; it may be abbreviated to V or H, and horizontal is assumed if omitted. The SPLIT command itself may be abbreviated to SP. For other forms of the SPLIT command defining complicated splits see below.
The SPLIT command will always leave the current buffer as one of the buffers inthe split list; if you are already in one mentioned in the list then it

takesno action but otherwise it switches to the first in the list.
When you want to revert to non-split editing the command to use is SPLIT NO.

4.4.1 Switching Between Windows

When you use split screen editing you will have several buffers displayed on the screen at the same time. The easiest way to switch between windows isto type DEL N; this switches to the next window in the split screen. The opposite function CTRL-N N switches to the previous window.
To switch to a particular buffer displayed you can usethe BUFFER, NEXTBUFF or PREVBUFF commands (after typing DEL C of course).For instance, if you had typed

```
        SP 1 2 3 H
```

and you were currently in the top window on the screen editing buffer 1, you could switch over to buffer 3 by typing DEL C BUFFER 3 {RETURN}. You will probably find it convenient to set up macros for switching easily to the firstfew buffers: for instance DEL 1 could switch to buffer 1, DEL 2 to buffer 2and so on.
You were probably just thinking 'What happens if I switch to a buffer that is not on the screen?', the answer being that the new buffer appears occupying thewhole screen and split screen editing is temporarily suspended. When, however,you switch back to one of the buffers that is part of the split you will go back to split screen editing with the original buffers displayed. You can, ofcourse, change the split at any time, or revert to no split by typing SPLIT NO.
If you type the SPLIT command alone (i.e. with no parameters after it), then the editor displays information about the current split (if any). This will show you how the available screen size has been divided to provide the windowsthat you have requested. You may have a terminal that is capable of changingits screen size to a greater width, usually 132 columns; when you use the TERMcommand to change the screen size the window sizes will be recalculated basedon the new area available.

4.4.2 More Complicated Splits

On certain occasions it may be useful to split the screen both vertically and horizontally to form square-shaped windows which may allow you to inspect your files more easily. This can be achieved by adding one extra parameter tothe SPLIT command, known as the 'split ways' parameter. For example, the command

```
        SPLIT 1 2 3 4 5 6 VERTICAL 3
```

means split up the screen into six windows, first splitting it into 3 verticalcolumns (the 'split ways' parameter is 3), and then splitting it furtherhorizontally to provide the necessary number of windows; in this case thewindows will be laid out as

```
        1 2 3
        4 5 6
```

Exactly the same effect could be achieved by

```
        SPLIT 1 2 3 4 5 6 HORIZONTAL 2
```

which would first split the screen horizontally into two and then vertically togive the requisite number of windows.
The buffers are always allocated to the windows by rows starting from the top left. If the number of buffers given is not divisible by the 'split ways' parameter then one or more empty windows will be left at the bottom right. Forexample

```
        SPLIT 1 2 3 4 5 VERTICAL 3
```

would result in windows laid out as

```
        1 2 3

        4 5
```

4.5 Safety Measures

If you try to delete a buffer which is tagged 'modified' you will be told that it has been modified and asked if it is 'OK to delete?' In a similar way, the Editor will query any attempt to overwrite a modified buffer when you issue an EDIT command. The 'modified' tags are cancelled when a fileis SAVEd or copied into another buffer by means of a LOAD command.

5. THE TIDY SUB-SYSTEM

The editor has a built-in system to help you with the task of clearing out allthe rubbish that builds up in your directories after a hard day's work at theterminal. It displays a list of the entries in the directory and allows you tolook at them and mark for deletion any that you do not want to keep. When you have finished you are shown a list of all the files you have marked fordeletion and asked whether you want to go ahead and clear them out. You canalso go down into any sub-directories and clean those up too.
This subsystem is normally run by using the CPL command TIDY; this performs some operand checking and then enters the editor. You are recommended to use this in preference to entering directly.
To clean up the current directory type

```
   TIDY
or
   ED -TIDY
```

The editor creates a file containing a list of all the entries in the current directory and enters the window editor to display it on the screen. The list normally shows all entries in alphabetical order, but you can change this by typing extra options:

TIDY -SORTM	Sort in date order.
TIDY @@.LIST	Show only files matching the given wildcard.
TIDY -REVERSE	Reverse the usual sorting order.

When you have the file list displayed on the screen you can move the cursor tothe line containing an entry you are interested in and use the commands:

DEL I	Inspect the entry. If it is a file it will be displayed on the screen in normal window-editor fashion. If it appears to be a binary file you will be asked whether you really want to display it.
DEL D	Mark the file to be deleted later. This merely overlays a letter 'd' on the first column of the line.

When you have typed DEL I and are displaying a file you can type any of:

DEL D	Go back to the file list and mark this file for deletion.
DEL L	Just go back to the file list display and do not mark for deletion. An '*' is put in column 1 of the display just to remind you that you have looked at that file. It will not be deleted later.
DEL I	Go back to the file list, do not mark for deletion, and then inspect the next entry on the list. This allows you easily to go down a list of files looking at each one in turn.

When you have finished with the directory listing you can type:

DEL F	Finish with this directory with the option of deleting marked files. If you have marked any files for deletion you will be shown a list of them and asked whether you want to go ahead and delete them.

DEL Q   Quit and do not delete any files.
To go down into a sub-directory place the cursor on the entry in the file listand type DEL I. The editor will display a new file list showing the contentsof this sub-directory and you can inspect and delete any entries as before.When you have finished in the sub-directory type DEL F or DEL Q as above toreturn to the higher level.
While using the tidy system you have all the usual editor commands available, so you can easily locate any files you are interested in. It is also possibleto actually change a file that you are inspecting but do not forget to save itbefore returning to the file list.

APPENDIX A SUMMARY OF WINDOW EDITING COMMANDS

DEL A 2.5	Convert all or part of the current line to UPPER case.
DEL a 2.5	Convert all or part of the current line to lower case.
DEL B 2.5	Break the current line into two at the current cursor position.
DEL C 2.2	Obey a line editor command.
DEL D 2.6	Delete marked text (see also DEL M) and copy to a specified buffer file.
DEL F 2.11	Copy the workfile to a permanent file and end the edit.
DEL I 2.5	Toggle between insert mode and overtype mode.
DEL J 2.5	Join the next line to the current line at the current cursor position.
DEL L 2.6	Loads text from a specified buffer at the current cursor position (see also DEL U, DEL D, DEL M).
DEL M 2.6, 2.8	Put a marker at the current cursor position for unloading (see DEL D, DEL U or text-formatting).
CTRL-N M 2.6, 2.8	Cancel the marker.
DEL N 4.4.1	Switch to the next window of the split screen.
CTRL-N N 4.4.1	Switch to the preceding window of the split screen.
DEL O 2.5	Inserts 12 blank lines above the current line.
CTRL+N O 2.5	Deletes any blank lines between the current line and the next line of text.
DEL P 2.5, 2.8	Mark the current cursor position.
CTRL+N P 2.5	Remove the currently active marker.
DEL Q	Leave the editor without copying the workfile to a permanent

2.11	file.
DEL R 2.9	Execute the next command a specified number of times.
DEL S 2.2	Saves the workfile in a filestore file without leaving the editor.
DEL T 2.7	Sets a tab stop at the current cursor position.
CTRL+N T 2.7	Clears the tab stop at the current cursor position.
DEL U 2.6	Copies marked text (see DEL M) to a buffer file.
DEL V 2.4	Move the cursor down 12 lines.
DEL W 2.3	Switch to window edit mode from input window mode or vice versa.
DEL X 2.2	Execute the last command again.
DEL Z 2.5	Delete the marked part of a line (see DEL P).
DEL + 2.5	Insert a blank line above the current line
DEL - 2.5	Delete the current line
DEL = 2.2	Redraw the current line
DEL " 2.5	Insert a copy of the current line.
DEL & 2.5	Insert one space at the current cursor position.
CTRL+N & 2.5	Delete one character at the current cursor position.
DEL space 2.5	Insert 20 spaces at the current cursor position.
CTRL+N space 2.5	Delete any spaces to the right of the cursor on the current line.
DEL / 2.5	Truncates the current line at the current cursor position.

DEL >	2.7	Move the cursor to the next tab position.
DEL <	2.7	Move the cursor to the previous tab position.
DEL '	2.8	Initiates text-formatting.
DEL (	2.10	Start definition of a macro.
CTRL+N (	2.10	Delete a specified macro.
DEL)	2.10	End definition of a macro.
DEL "uparrow"	2.4	Move the cursor to beginning of file.
DEL "downarrow"	2.4	Move the cursor to end of file.
DEL "rightarrow"	2.4	Move the cursor to one past the last character on the line.
DEL "leftarrow"	2.4	Move the cursor to the beginning of the line.
DEL ^	2.4	Move the cursor up 12 lines.
RETURN	2.4	Move the cursor to the beginning of the next line (but see also the line editor MODE INDENT command).
HOME	2.4	Move the cursor to the top left hand corner of the screen.
BACKSPACE	2.4	Move the cursor one place to the left.
"cursor arrows"	2.4	Move the cursor one place in the direction shown on the key.
TAB	2.4, 2.7	Move the cursor to the next tab position.
BACK TAB	2.4, 2.7	Move the cursor back to the previous tab position.

<u>FUNCTION KEYS</u>
Many terminals have a row of function keys along the top of the keyboard or to one side; it will usually be possible to use these keys to perform certain common window editor commands by a single keystroke. Sincedifferent

terminals have different numbers of function keys it is notpossible to say exactly what will be provided, but you can find out bytyping the line-editor command HELP SUMMARY. On the Televideo and Merlin terminals in use at Sheffield the set is as follows:

Key	Without Shift	With Shift
F1	Exit to Line Ed	Quit
F2	Redraw Screen	Redraw Line
F3	Help	Repeat
F4	Cursor Down half screen	Cursor Up half screen
F5	Break Line	Join Line
F6	Mark Line	Mark Char
F7	Unload Lines	Delete Lines
F8	Load Lines	Duplicate Line
F9	Save	File
F10	Switch WI <-> IW	Insert Mode
F11	Obey Command	Obey Command again

APPENDIX B SUMMARY OF LINE EDITING COMMANDS

These directives must be followed by the RETURN key in the usual way; permittedshortened forms are shown by capital letters.

ALphabet 3.12	Specify the alphabetic characters.
Append string 3.5	Appends string to the end of the current line.
Bottom 3.4	Moves pointer to the bottom of the workfile.
BRief 3.12	Suppresses verification output.
BUFFer n 4.1	Switches to file buffer n or the buffer matching a string. If no n is specified, a list of your buffers is displayed.
CEntre 3.9	Centres the text on the current line between the margins specified by the STYLE command.
Change/string1/string2/G end 3.5	Replaces string1 by string2. If G is present, replaces all occurrences on specified line(s); if G is omitted, only first occurrence is replaced. The lines to change are determined by end which may be n, -n, *, M for 'next n lines', 'previous n lines', 'throughout file', or 'to marker' respectively.
COminput filename 4.2	Reads and obeys lines of editor commands from the given filename.
DELBuf n1 n2... 4.3	Deletes buffers n1, n2....
Delete n Delete TO string 3.6	Deletes n lines or up to but not including the line containing the specified string.
DUnload file n DUnload file TO string 3.7	Deletes n lines, or up to but not including the line containing the specified string, and copies them to an editor buffer or Primos file.
EDit filename 3.2, 4.1	Copies specified file into workfile for new edit. If no filename is specified starts new file in input mode.
EXPand 3.5	Convert any tabs in current line to equivalent spaces.

Erase character 3.2	Makes the specified character the ERASE character.
EXtract buffer col1 col2 options 3.14	Extract a vertical column of text from the current buffer for editing separately.
FILe filename 3.19	Copies workfile to specified file.
Find string Find(n) string 3.4	Moves the pointer to the first line with the specified string starting in column 1 or column n.
FName filename 3.2	Changes the 'current name' of the edited file to the specified name. If no name is specified, prints out the current name.
Gmodify arglist 3.5	Changes current line according to list of directives which may or may not be separated by spaces. See section 3.5 for a list of directives.
Help 3.12	Enters the editor internal HELP system.
IB string 3.6	Inserts the specified string as a new line immediately before the current line.
Insert string 3.6	Inserts the specified string as a new line immediately following current line.
IW 3.3	Switches to input window mode.
Kill character 3.2	Makes the specified character the KILL character.
LInesz n	Reports any lines of length n characters or more.
LOAd file 3.7, 4.3	Copies the specified file into the workfile; the file may be specified by name or as a file buffer number.
LOADState filename 4.2	Reads editor commands from the given file to restore the editor to the state saved by the SAVESTATE command.
Locate string Locate(n) string	Moves the pointer to the next line which contains the specified string on or after

3.4	column 1 or column n.
MACClear 2.10	Clears all current window editor macros.
MACLoad filename 2.10	Loads the window editor macros from the specified file.
MACPrint maclist 2.10	Displays the specified macros, or all macros if maclist is omitted.
MACSave filename 2.10	Saves all current window editor macros in the specified file.
MArk n 3.11	Places a 'marker' at the specified line. The marker remains set until a different marker is set, or until MARK is called with n zero. If n is omitted, a marker is set at the current line. Markers are used in with commands such as CHANGE and TEXT.
MErge buffer col1 col2 options 3.14	Merge the given buffer side-by-side with the current buffer.
MODE 3.13	Displays current MODE and other settings.
MODE Backup ncopies MODE NoBackup 3.13, 2.11.1	Activates/deactivates taking of backup copies before overwriting file.
MODE CHain list MODE NoCHain 3.13, 4.3.1	Activates/deactivates chaining together a set of buffers so that they are treated as one file by commands such as LOCATE.
MODE Ckpar MODE NoCkpar 3.13	Activates/deactivates parity checking.
MODE COlumn MODE NoCOlumn 3.13	Activates/deactivates the column indicator at the start of input mode or before any printing directive.
MODE COMo MODE NoCOMo 3.13	Activates/deactivates allowing como file to stay open while in window editor.
MODE COUnt n1 n2 n3 mode MODE NoCOUnt 3.13	Activates/deactivates counter symbol. Negative counting is allowed.
MODE Direct MODE NoDirect	Activates/deactivates editor operation suited to directly-connected terminals on

3.13	lightly-loaded machines.
MODE DISp MODE NoDISp 3.13	Activates/deactivates display of buffer number and filename whenever buffer changed.
MODE INdent MODE NoINdent 2.7, 3.13	Used with window editing, and causes the RETURN key to return the cursor to the same starting position as the line above.
MODE Info MODE NoInfo 3.13	Activates/deactivates info editing mode
MODE Lon MODE NoLon 3.13	Activates/deactivates printing of logout notification messages during window edits.
MODE Margin MODE NoMargin 3.13	Activates/deactivates word-processing input margin mode.
MODE MSg MODE NoMSg 3.13	Activates/deactivates printing of messages during window editing.
MODE NUmber MODE NoNUmber 3.13	Activates/deactivates the printing of line numbers.
MODE Prompt MODE NoPrompt 3.13	Activates/deactivates display of input and edit prompts.
MODE Quiet MODE NoQuiet 3.13	Activates/deactivates window editor quiet mode.
MODE SAfety MODE NoSAfety 3.13	Activates/deactivates safety checking when writing to filestore.
MODE Semi MODE NoSemi 3.13	Activates/deactivates ; as newline in input mode
Modify/string1/string2/G end 3.5	As CHANGE, but with column alignment preserved.
MOVe buffer1 /string/ MOVe buffer1 buffer2 3.15	Moves specified string or buffer2 into buffer1. EDLIN is a line buffer containing the previous edit directives, INLIN is a buffer containing the current workfile line.
NEW filename	Switches to first empty buffer and loads the

4.1	given file into it for editing.
Next n.m 3.4	Moves pointer n lines, and positions it at the mth column. If n is negative, movement is backwards. If m is omitted, the pointer is positioned at the first character.
NEXTBuff 4.1	Switch to next buffer containing text.
NFind string NFind(n) string 3.4	Moves the pointer to the next line which does not contain the specified string starting in column 1 or n.
NLocate string NLocate(n) string 3.4	Moves pointer to the next line which does not contain the specified string on or after column 1 or n.
OBey 'comm1;comm2...' 3.16	Obey a line of editor commands after variable substitution.
OOps number 2.5.1, 3.2	Reinstates 'current line' as it was before current alterations, or inserts old copy of line (window editor action).
Overlay string 3.5	Superimposes the specified string on the current line.
POint n.m 3.4	Moves pointer to column m of line n.
PPrint m n 3.2	Prints m lines before, and n lines after, the current line.
PREvbuff 4.1	Switch to previous buffer containing text.
Print n Print buffer Print All 3.2, 3.15	Prints the next n lines (or previous n lines if n is negative), or prints contents of the specified line buffer, or all buffers.
PSymbol 3.12	Prints a list of the special characters. Any invisible characters will be shown as octal ASCII codes.
PUnct 3.12	Sets the characters to be treated as punctuation.
QF 3.19	Leaves the editor without saving the edited version of the file, and without asking if you really want to go.

Quit 3.19	As QF but asks if you really want to 'quit'.
RELoad 4.1	Reload a fresh copy of the file being edited.
REPlace/string1/string2/ci 2.5.2	Window editor search and replace command. (May only be used via DEL C from within the window editor.)
Retype string 3.5	Deletes the current line, replacing it with the specified string.
SAve filename 3.2	Saves the edited version of your file, but leaves you in the editor to continue editing.
SAVEAll 4.2	Saves all the file buffers.
SAVEState filename 4.2	Saves all file buffers and writes editor commands to the given filename, so that the editor state can be restored by the LOADSTATE command.
SIze 4.3	Display sizes of files stored in all editor buffers.
SPEll 3.17	Run the spelling checker on the text in the current buffer.
SPlit n1 n2... orientation n 4.4	Tells the window editor to initiate split screen editing, listing the numbers of the buffers to be included in the split, whether it is Horizontal (default) or Vertical, and also the split-ways parameter if necessary.
STAts 3.9	Counts words and lines in part or all of a document.
STyle lm rm pi pb mode 3.9	Used for text formatting to specify left margin, right margin, paragraph indentation, blank lines between paragraphs and adjust mode (Adjust or Fill).
Symbol function character 3.9, 3.12	Resets the specified character as a special character as defined by function.
TAbset t1 t2 ... 2.7, 3.8	Sets tab position. Up to 30 positions may be established.
TEXt n	Formats file according to most recent STYLE

3.9	specification down to the next marker (see MARK). Any lines not starting in column n will be assumed to start a new paragraph. If n is omitted, 1 is assumed; n is not used in automatic formatting.
Top 3.4	Moves pointer to the top of the file.
Unload file n Unload file TO string 3.7, 4.3	Copies n lines, or up to but not including the line containing the specified string, to the specified file. The file may be specified by name or as a buffer number.
Verify 3.11	Activates verification output.
VERSion 3.12	Prints the editor version number.
Where 3.2	Prints the current line and column number, and buffer number.
WI 3.3	Switches to window mode.
Xeq buffer 3.15	Executes the contents of the specified buffer. If no buffer is specified, executes last command again.
* n	Repeats line of editing directives n times.
! command 3.12	Obeys the specified Primos command, after expanding abbreviations if enabled.

In addition to the above, a directive may be built up of any of the following elements to achieve the outcome indicated.

P	for	Print
D	for	Delete
U	for	Unload
B	for	Backwards
C	for	Case independence
I	for	Identifier type
N	for	Not

The composite command is then followed by:

(a) F or L for a FIND or LOCATE type string specification, with a (column) number in parentheses if required

(b)

string	if it includes U
filename TO string	if it does not include U

APPENDIX C TERMINAL CONTROL KEYS AND SPECIAL CHARACTERS

Special Keys and Characters for Use at All Times

In the list below, where two keys are pressed simultaneously, they are shown as a+b, e.g. CTRL+E etc.

RETURN	Causes the computer to take action on the line you have typed.Until you press RETURN you can correct the line using your erase and killcharacters. Window editor commands are not followed by RETURN.
BRK	Produces a 'break-in'; it is used to interrupt the Prime in whatever it is doing, and to bring control back to the keyboard. Prime terminalsonly.
CTRL+P,B	Produces the same effect as the BRK key. Network terminals only.
CTRL+S	Suspends the activity of the terminal when it is outputting information (e.g. a file listing, program results etc.).
CTRL+Q	Re-activates the terminal after it has been stopped by CTRL+S.
CTRL+X	Cancels the action being prompted for; e.g. when typing a filename CTRL+X cancels the operation that prompted for the filename.
BACKSPACE	Erases the previous character of the current line.
?	Kills the whole of the current line typed so far.

Special Characters for Use with the Editor

;	Is equivalent to a 'newline' character when in line editorinput mode.
^	May be used to precede BACKSPACE or ? (or itself) to givethese characters their face value.
!	Is used in connection with string specifications for FIND and LOCATE etc. to mean 'any character'.
#	Is used in the same circumstances as ! and means'any number of spaces or no spaces'.