User Manual

IMP-9 EDITOR

Temporary Restriction

The version of this program dated 29/10/70 converts all lowercase letters to uppercase on input.

C.Whitfield,
Computer Science Dept.
7, Buccleuch Pl.,
Edinburgh University.

General Characteristics

This document describes the command repertoire and operation of an interactive context editor. The program is written in IMP and all I/O is via logical streams using standard IMP subroutines. The Editor is therefore, completely device independent, this independence extending even to the command stream.

Commands are represented by alphabetic names which for the more frequently used commands are single letter mnemonics. In principle, any number of commands may be concatenated to form a command string.

A distinction is made between line locating commands and within-the-line editing commands. Line location is either by moving backwards or forwards a specified number of lines, searching forward for a specified text string or, subject to some restrictions, by specifying a line number, when the lines in the file are treated as if sequentially numbered upwards from 1.

All Editor commands take effect at a point in the file marked by a movable file pointer. This pointer marks the point in the file reached by the last command executed ; it is initially set to the start of the file.

The command language permits nested loops with associated loop counts to be set up within one command string ; exit from a loop occurs either when some enclosed command cannot be executed or when the the loop count is exhausted.

Both secondary input and output streams are available. The former can be read selectively and parts of it included in the file being edited. A paged, line-numbered listing of the output file can be requested on the secondary output stream or, subject to the same restrictions as location by line number, selected parts of the file can be so listed.

(over)

On console input, single characters, whole lines and whole commands may be cancelled to correct typing errors.

It is not possible to interrupt a command once execution of it has started except by forcing a reload of the IMP Executive.

Section I

This first section of the manual describes a subset of the Editor's instructions. Make sure that you are familiar with them before going further since then at least you will have something to fall back on in case of difficulty.

In the event of difficulty of any sort with the IMP-9 Executive or the Editor, keep the teletype printout ; especially important is the piece immediately preceding the failure as it may give a clue as to the cause.

Loading the Editor

If the IMP-9 Executive is not loaded and waiting for a command you must first load it. See the IMP-9 manual for details of how to do this.

The Editor operates on IMP logical I/O streams which are connected to real files specified when it is loaded ; if not explicitly specified a default value is supplied automatically. Thus for the simple case of editing a DECtape file called, say, 'FILE' recorded on a tape mounted on unit number ONE we need only type:

```
EDIT FILE
```

When correctly loaded, the Editor will type out 'IMP EDIT' followed by a date.

See Appendix 1 if it does not.

For full details of I/O stream usage, see Appendix 2

Teletype Conventions

The IMP-9 Executive permits single characters and whole lines to be deleted and retyped if a mistake is made on console input. 'Rubout' deletes single characters, 'control+U' deletes whole lines. See the IMP-9 Manual for more detail. In addition, 'control+D' may be used to cancel a complete command even if it extends over several lines ; this feature is specific to the Editor.

When ready to accept a command, the Editor types a right angle bracket(>) to signal this fact to the user. 'Control+D' causes a new prompt to be issued.

Basic Search Command

This is of the form:

F <quote><text string><quote>

Find initiates a search for the specified text, starting with the first character but one after the movable file pointer(FP) and continuing until either an occurrence of the text is found or end-of-file reached ; if the search is successful, FP is set to the first character of the text. Because the first character after FP is skipped before starting the search, successive calls on 'F' locate successive instances of <text string>.

See Appendix 4 for concise definitions of <quote> and <text string>.

Example

```
>F 'FRED'
```

Find the next occurrence of 'FRED' in the file, searching left to right and top to bottom ; if FP is already set to an occurrence of 'FRED', skip it and take the next one.

Basic Commands Requiring a Text Parameter

These are of the form:

<command letter><quote><text string><quote>

Spaces are ignored except in <text string>.

<command letter> is:

A set FP after specified text

B ..... before .....

D set FP to, and delete specified text

U delete from current position of FP up to but excluding specified text

I insert specified text before FP

The text search operates from left to right along the current line starting from the current position of FP. Though the search must start on the current line it may continue onto the next line(s) to complete the matching of strings containing one or more newline characters.

FP is only moved if the search is successful.

Example

```
>D 'FRED' I /TOM/ F.JIM. A/JIM/ I.MY.
```

Starting at the current position of FP, search left to right along current line to the next occurrence of 'FRED' ; abandon the command and type diagnostic if there is no such occurrence. Having found it, set FP to it then delete it leaving FP pointing to the following character.

(over)

Insert 'TOM' to the left of FP.

Search left to right, top to bottom through the file to the next occurrence of 'JIM' ; exit with message if there is no such occurrence otherwise set FP to the first character after 'JIM'.

Insert 'MY' to the left of FP.

Note

a) That the use of '.' and '/' as quote marks in place of the more obvious ''' is governed purely by expediency ; the keys carrying these characters on a teletype keyboard just happen to be conveniently placed. To aid legibility however ordinary quote marks will be used throughout the rest of this document.

b) To save some typing, a dash(-) can be used to stand for the last text parameter typed in ; thus the above example could be shortened a little to:

```
> D'FRED' I'TOM' F'JIM' A- I'MY'
```

Basic Commands Requiring a Numerical Parameter

These are of the form:

<command letter><number>

Spaces are ignored except as a number terminator.

<command letter> is:

M move forward or backward the number of lines specified ; FP is set to the start of the target line. A positive parameter means forward, a negative one, backward.

P print on the teleprinter the specified number of lines starting with the current line. If more than one line is printed, FP is moved to the start of the last one printed otherwise it is not moved. The position of FP is marked by an up-arrow(↑) which is suppressed if it is at the beginning of a non-null line.

K kill the specified number of whole lines starting with the current one. FP is moved to the start of the line after the last one removed.

P and K are not defined for a negative parameter which will therefore be monitored.

<number> is a signed or unsigned non-zero integer taking a default value of 1 ; '*' represents an indefinitely large value. See Appendix 4 for a more precise definition.

(over)

Examples

To aid description in the following examples a line is at some point taken to be line N and other lines thereafter referred to as N+1, N+2 etc. This numbering is fixed at the time N is assigned ; for example, N+1 and N+3 are consecutive lines if N+2 is removed.

>P 5

take current line as N ; print lines N to N+4 inclusive on the console and leave FP set to the start of N+4.

>MP

take the current line to be N ; set FP to the start of N+1 then print N+1.

>F'3:' M-2 K3 M- P2

search through the file for the next occurrence of '3:' ; take the current line as line N. Set FP to the start of N-2 ; destroy N-2, N-1, and N leaving FP set to the start of N+1. Go back one line, setting FP to the start of N-3 ; print lines N-3 and N+1 which are now contiguous, finally leaving FP at the start of N+1.

Basic Commands Requiring No Parameter

These are of the form:

<command letter> or πCLOSE

<command letter> is:

T go to the top of the file

L set FP to the left-hand margin, i.e. to the start of the current line.

R set FP to the right-hand margin, i.e. to end of current line.

πCLOSE outputs the final edited version of the file then reloads the IMP Executive. Note that the original is completely intact until this command is given ; editing may be terminated at any time by reloading the Executive with no risk at all of destroying the original though all editing done on that run will be lost.

Input Mode

Normally, anything typed on the console is treated as a command and an attempt made to execute it. To facilitate making large insertions, it is possible to switch the Editor into a state in which all console input is treated as text to be inserted in the file being edited.

The parameterless command '::<newline>' causes this switch to input mode to take place. Note that it is obeyed strictly in sequence and may be appended to any command string.

To switch back again, type '::<newline>' at the start of a line.

These commands are arranged so that only complete lines can be inserted in this way ; FP is moved to the start of the current line and the insertion made immediately before it, i.e. above the current line.

Points to Note

1) When in input mode, the only editor command which is recognised as such is '::<newline>' ; all others are merely text to be inserted in the file.

2) FP is moved to the beginning of the current line.

3) Text goes in above the current line ; so, the last line inserted is not the current line, but the line above.

4) If in the middle of a large insertion you wish to alter the line just typed in, change from input to edit mode, go back a line, make the

(over)

required alterations, move forward a line then switch back to input mode. As Editor commands this appears:

```
THIS IS THE LATEST LINE TYPED IN
::                                  switch to edit mode
>M-                                 move back a line
> ? ......                          make required changes
>M::                                move on a line then re-enter input mode
THIS IS THE NEXT LINE TYPED IN
```

5) 'Kn::' provides a convenient way of changing a number of lines ; the n lines removed using 'K' are replaced by whatever is typed in following '::<newline>.

Messages

The Editor operates by decoding a whole command string then executing it ; messages may be printed either during decoding or during execution. The latter are preceded by an asterisk ; the former are not.

Messages Printed While Decoding

These are invariably error messages ; the whole command string is ignored and another requested.

All but one of the messages printed are self-explanatory, this one appearing whenever a non-existent editing function is requested. It is of the form:

'FN= <name of function> ?'

The most usual cause is a faulty quote mark in a preceding text string. For example:

>B.X = 37.2*Y + Z.

would take B's parameter to be 'X = 37' then try to decode '2*Y + Z' as the next command. The message 'FN= 2 ?' would be typed. Presumably the intended string was 'X = 37.2*Y + Z'.

Messages Printed During Execution

These in general only inform the user what is happening ; whether or not this constitutes an error must depend on what he or she is trying to do.

The possible messages are as follows:

1) '*<text string> NOT FOUND'

The given string was not found between the current position of FP and

(over)

the end of the line. FP is not moved and the command is abandoned at this point.

2) '*OFF TOP'

You have run off the beginning of the input file ; this is only possible using 'M-'. The command is abandoned at this point leaving FP set to the beginning of the file.

3) '*EOF <n>'

You have run off the end of the input file. The command is abandoned at this point. <n> will be explained in the next section and should be ignored meantime. FP is left at the very end of the file.

3) '*.1 COMPLETE' or '*.2 COMPLETE'

The Editor uses two IMP scratch files called '.1' and '.2' as work space. Should a serious error be made or a system failure occur, it is possible to restart from the point at which the latest '*COMPLETE' message was produced. See Appendix 3 for details of how to do this.

4) '** IMP EDIT: <message> <number>/ <number>'

An irrecoverable error has occurred. At this stage the only likely <message> is 'ASL EMPTY' which would result from trying to edit a file containing lines which are grossly too long. The Editor run is terminated and the IMP Executive reloaded. No output file is produced.

See Appendix 5 if <message> is other than 'ASL EMPTY'.

Section II

The Editor has a number of features which for simplicity's sake were left out of the preceding section. Some more editing commands, a means of repeating all or parts of a compound instruction and the commands available for manipulating a secondary, read-only, input file are described here.

Repetition

Thus far, command strings consist of one or more simple commands with no limit, in principle at least, to the number which can be concatenated. A fairly obvious extension is to provide some means of setting up loops which in turn requires that the condition(s) under which looping terminates must be defined.

Loops can only be set up within one command string ; it is not possible, for example, to execute repetitively the last three command strings typed in.

The scope of a loop is specified by parentheses ; the loop count, i.e. how often the loop is to repeated, by a positive numerical parameter, as defined in the last section, after the closing bracket. Parentheses, with their associated loop counts may, in principle, be nested to any depth.

An error condition arises within a loop if one of the enclosed simple commands cannot be executed ; for example, an attempt might be made to move further down the file when already at the end. Whenever such an error occurs, the immediately enclosing loop is terminated without regard to its loop count ; if there is no such loop, the command is abandoned.

The rule for any loop is:

'Repeat this loop until either the loop count is exhausted or one of the enclosed simple commands fails ; when this happens skip out one level of bracketting and continue.'

N.B. Throughout this manual, the phrase 'the command is abandoned' should really be qualified by 'providing it contains no parentheses' ; some errors definitely abort the command string, others just skip one level of bracketting. See Appendix 5 for details.

Some More Commands

These are as follows:

1) C <quote><string1><quote><string2><quote>

change <string1> to <string2> ; has exactly the same effect as 'delete <string1>, insert <string2>' but is less trouble to type.

2) N <quote><text string><quote>

put a new line in place of the current one. i.e. remove the current line and replace it with the text string supplied.

3) X <number>

exit the number of levels of bracketting specified by <number> ; 'X1' skips one level, 'X2' skips two and so on. 'X*' causes the command string to be abandoned completely.

4) S <number>

step FP forwards(+ve) or backwards(-ve) the number of characters specified. The command is abandoned and the message '*EOL' printed if the movement specified would take FP out of the current line.

5) πLD

left delete :- delete that part of the current line which lies to the left of FP.

(over)

6) πRD

right delete :- delete that part of the current line which lies to the right of FP.

7) πU <quote><text string><quote>

Same as U except that its scope is not restricted to the current line with the attendant risk of destroying a large part of the file if a mistake is made.

8) πNL <number>

Insert <number> newline characters behind FP.

9) πLISTCLOSE

Typed as an alternative to πCLOSE, it causes a line-numbered listing to be output in addition to the usual edited file.

10) ! <number>

Execute the last command string again the specified number of times.

Note

Multi-letter command names must be separated by a non-alphabetic character from following single-letter mnemonics ; a space is sufficient. For example, 'πRD P' is acceptable but 'πRDP' is not.

Examples

1) Print the current line and every tenth line thereafter for the next fifty lines:

```
>P( M10 P )5
```

2) Change '1' to '2' in the fifth source statement on a line of IMP program text:

```
>( A';' )4 C'1'2'
```

3) Change calls on 'PRINT' to calls on 'WRITE' throughout an IMP program. The latter has two, not three parameters.

```
>T( F'PRINT(' C-'WRITE(' A',' B- U')' P )*
```

In this last example, note the use of:

a) 'T' to make sure that FP really is at the top of the file before starting.

b) 'P' to print out any line that is altered.

c) 'PRINT(' rather than just 'PRINT' so that for example, 'PRINT SYMBOL' won't be changed.

d) The use of '-' to save retyping parameters.

e) Spaces to make the command string more readable and easier to type correctly.

Note that the string search does not ignore spaces so that although for example, 'PRINT(' and 'PRINT (' have the same meaning in IMP, they are different strings of characters and will be treated as such by the Editor.

Two Problems

A powerful feature of the notation described is that by setting up a loop with a very large loop count('*') and moving FP backwards at some point within this loop, it is possible to make repeated passes over a section of the input until some condition is met ; on the other hand, it is equally possible to set conditions that will never be met and put the Editor into a non-terminating loop.

So far as I know there is no general way of distinguishing commands that will cause trouble from those that will not.

The second snag is that since the success or failure of individual commands may be used to control branching within a bracketted command string, failure of any individual component of such a command string does not automatically imply failure of the whole string in the way it does with unbracketted commands. There are consequently difficulties in producing adequate diagnostics for bracketted commands.

Secondary Input

Often it is convenient to edit portions of one file into another, for example, to use routines from one program in another. To make this possible, the Editor allows a second input file to be specified when it is loaded ; this file can be read selectively, but not written to.

For example, by using:

EDIT FILE1, FILE2

parts of FILE2 can be incorporated in FILE1 but not vice versa. The output file produced will be called FILE1.

The *EOF Message

In the last section, a diagnostic message of the form:

*EOF <n>

was mentioned but the significance of <n> not explained.

<n> is a single digit identifying the input stream on which the end-of-file condition exists ; '1' denotes the primary input, '2' the secondary input.

Manipulating the Secondary Input Stream

The properties of the secondary input stream are very similar to those of 'input mode' described in the last section ; FP is moved to the start of the current line, whole lines are inserted above the current line and so on. The main difference is that instead of typing in the insertion itself on the console, you must tell the Editor how much to take and where in the secondary input file to take it from.

The commands available are:

1) πSKIP <number>

skip over the specified number of lines in the secondary input file.

2) πGET <number>

get the specified number of lines from the secondary input file and insert them above the current line.

3) πSKIPTO <quote><text string><quote>

skip up to and including the next line fetched from secondary input which contains the specified text. Print the last line skipped on the teletype.

4) πGETTO <quote><text string><quote>

get up to and including the next line fetched from secondary input which contains the specified text and insert these lines above the current one. Print the last line inserted on the teletype.

(over)

5) πREWIND

rewind the secondary input file if this is possible ; if the file is on an external device you will also have to rewind the paper tape or restack the cards yourself.

Section III

To facilitate using the Editor in a non-interactive environment with previously prepared command strings, line location by number and selective listing of parts of the file can be had in exchange for the ability to move towards the beginning of the file and have newlines in <text string>'s.

The secondary input file can still be rewound.

Onepass Mode

The commands associated with this feature are:

1) πONEPASS

Disable 'T' and 'M-' and make available the following commands. 'πONEPASS' can only appear as the very first command of an Editor run.

2) πLINE <number>

Considering the lines in the file to be implicitly numbered sequentially upwards from 1, set FP to the beginning of line <number>. Attempts to move backwards will be monitored.

3) πAT <number>

Set FP at the beginning of line <number> as in (2) and also list all intermediate lines.

4) πNOLIST

Suppress all file listing except that produced by 'πLISTCLOSE'. This command can only be given at the beginning of an Editor run. It is irrevocable.

Example

πONEPASS	enter onepass mode
πLINE 150	skip to line 150
πAT 155 C'SUM'TOTAL'	list 150-154, change 155
πAT 160 ::	list 155-159, insert above 160

(over)

```
      TOTAL = TOTAL + ITEM
::
πAT 170 RI ' ; X = Y'          list 160-169, append to 170
M 5                            move 5 lines, list 170-174
πCLOSE                         list 175, close file
```

Note

The Editor prints all commands executed on the 'diagnostics' stream. If it is necessary to use one device for this and also for the selective listing produced by 'πLINE'/'πAT' a rather messy printout results. This can be avoided by suppressing the the selective listing with 'πNOLIST' and terminating with 'πLISTCLOSE'

Onepass Errors

While it is possible, because of the device independent properties of the IMP-9 operating system, to run the Editor interactively in 'onepass' mode, it is nonetheless true that 'onepass' is of more use in a non-interactive environment. Failure conditions are therefore treated differently than in 'normal' mode.

If an error of any sort occurs in 'onepass', the usual diagnostic is followed by:

'**IMP EDIT: <message> <n1>/ <n2>'

The IMP Executive is then reloaded.

<n1> is the number of the current line or the line-being-sought on the input file depending on whether the failure occurs during decoding or execution of a command.

<n2> is the line number of the current line on the output file.

<message> may be:

1) CONTEXT?

'πONEPASS' or 'πNOLIST' occurred partway through an Editor run.

2) DECODE?

Command could not be decoded.

3) EXECUTE?

Command could not be executed.

4) πCLOSE MISSING

The command stream went 'end-of-file' while in 'edit' mode.

(over)

5) :: MISSING

The command stream went 'end-of-file' while in 'input' mode.

6) ASL EMPTY

The input file contains line(s) which are grossly too long.

This diagnostic can also occur in 'normal' mode.

Appendix 1

The following is intended as a check-list for users who have difficulty in loading rather than in operating the Editor program. As such, it is more concerned with the IMP-9 operating system, with which the reader is assumed to be unfamiliar, than with the Editor itself. The details of this check-list are specific to the DEC PDP-9.

1) Check that the teletype is on 'line' not 'local' and that the IMP-9 system tape is mounted on unit 8(logically 0) which should be 'remote' and 'write enabled'.
Set the left-hand switches to 20(octal).
Put the short paper tape bootstrap in the reader then press 'I/O RESET' followed by 'READ IN'.
Go to (2) if 'IMP9' is printed on the console.

Is the teletype on 'line'?
Are the left-hand switches set to 20?
Did you put the paper-tape bootstrap in the reader the right way round? (with the narrow edge to the back and arrows pointing left)
Is unit 8 'remote' and carrying the IMP system tape?

Try again and if absolutely no response is evoked, look for help.

2) Call the Editor by typing an appropriate command string, in the simple case:
EDIT <file name>
Go to (7) if characters typed on the console are not echoed.

(over)

Go to (3) if the IMP Executive accepted your command string.

The format of your command must be wrong. Try again twice in case the teletype is producing faulty characters. Complain if it turns out to be the teletype ; it won't fix itself!

3) Go to (4) unless a message was typed of the form:

DTu NOT READY

Normally, the Editor uses unit 1 for the input file and the system unit for work space.

Taking u from the message, is unit u both 'remote' and 'write enabled'?

Ready the unit then type 'return'.

4) Go to (5) unless the message was typed:

'x NOT ON DTu' , where x is '1' or '2'

The unit selector switch on the deck carrying the system tape has been altered.

Set the system unit selector switch to 8 and go to (1).

5) Go to (6) if 'IMP EDIT', followed by a date, followed by '>' on a newline was printed out.

The Editor program was loaded incorrectly.

Reload the system and try again ; you may have to put in the paper tape but not necessarily so ; try typing 'alt mode' twice.

If you get repeated failures, complain. The system tape may be

(over)

corrupted and it won't fix itself either.

6) Go to (7) if the 'program stop' light (top left-hand corner) on the computer front panel is 'on'.

The Editor is loaded and waiting for a command.

7) The computer is not running. This may be the result of faulty loading or, less likely, a machine error or corrupted system tape. Reload the system using the paper tape bootstrap and try again. If the failure occurs repeatedly, look for help.

Appendix 2

The Editor communicates with its user via three input and three output streams. Two more streams are used but are not directly accessible by the user ; they are connected to the two scratch files, '.1' and '.2', which are used to store intermediate versions of the file as editing proceeds.

The use made of these six user-accessible streams, their default device assignments and their allocation to the IMP operating system's logical I/O streams is given below.

input	use	default
0	user commands	TT
1	primary input file	TT
2	secondary input file(read only)	N
output		
0	program generated diagnostics	TT
1	edited output file	=in1
2	secondary output(line-numbered listings)	LP

Appendix 3

As is described in Appendix 2, the Editor uses two IMP scratch files as work space in which to store intermediate versions of the file being edited. Whenever one of these intermediate versions is complete and generation of the next one started, a message is output of the form:

'*xx COMPLETE'

where 'xx' is either '.1' or '.2'. The meaning is that '.1' or '.2', whichever applies, holds the intermediate version of the file extant at the time the message is printed. This information can be used to restart from that point rather than the very beginning should a system failure or serious user error subsequently occur.

The following example assumes that the Editor was loaded by typing:

EDIT FILE

If the last '*COMPLETE' message before the failure referred to '.1' transfer it to '.2' using the IMP-9 Executive. To do this type:

T .1/.2

Obviously if it referred to '.2' this step is unnecessary.

Load the Editor by typing:

EDIT .2/ FILE

The above procedure will allow editing to continue, the final result when the file is closed being the same as if no break had occurred, but it is absolutely essential that the last '*COMPLETE' message is used and that when reloading the Editor, '.2' appears in the command.

Note that until the 'close' is given, the original version of 'FILE' is intact, irrespective of whether this restart procedure has been used or not.

Appendix 4

Various abstractions are used in the manual and loosely or implicitly defined at the first point of use. More precise definitions are given here.

1) A line of text is a string of characters other than newline terminated by a newline character ; the string may be null. Editing is done with respect to a movable file pointer(FP) which is used to keep track of the point currently reached in the file. The line containing FP is referred to as the current line.

2) <QUOTE> is implicitly defined as the first character, ignoring space and newline, after the function letter of a command requiring a text parameter ; it may be any non-alphanumeric except '(', ')', '*', 'π' or '-'.

3) <TEXT STRING> is a string containing any character except <quote>. It must not be longer than one full line but may contain newline characters. Maximum line length varies between implementations but will be 80-100 characters at least. If secondary input is not in use, lines up to 50 percent longer than the set maximum will be accepted.

4) <NUMBER> is a signed or unsigned decimal integer. Sign or digits or both may be omitted when they have default values of '+' and '1' respectively. Thus '1', '+1', '+' and <null> are equivalent as are '-' and '-1'.

Zero is unacceptable and will not be recognised as a <number>.

'*' represents an indefinitely large number ; '-*' is acceptable.

Appendix 5

Recoverable Errors

message	comment
decoding errors	
INSTR TOO LONG	> 300 characters
TEXT TOO LONG	> line length
FN='....' ?	quote mark missing?
QUOTE='....' ?	illegal quote character used
! IMPOSSIBLE	last command no longer available
- IMPOSSIBLE	last text parameter not available
-VE NO. ?	illegal -ve. parameter
BRACKETS ?	unbalanced parentheses
execution errors	
*'....' NOT FOUND	A,B,C,D or U failed
*EOF n	end-of-file on stream n
*EOL	S±n with n too big
*LOOP TRAP	non-terminating loop detected
*OFF TOP	M- at top of file ?

NOT FOUND,EOF and EOL skip out one level of bracketting and continue; LOOP TRAP and OFF TOP cause the command string to be abandoned completely.

(over)

Irrecoverable Errors

These are of the form:

'**IMP EDIT: <message> <n1>/<n2>'

<n1> is the line number of the current line or line-being-sought on the input file and <n2> the current line number on the output file at the time the error occurred.

message	comment
ASL EMPTY	excessively long input line(s)
CONTEXT?	'$BATCH' or '$NOLIST' not at beginning
DECODE?	onepass decoding error
EXECUTE?	onepass execution error
$CLOSE MISSING	command stream EOF in 'edit' mode
:: MISSING	command stream EOF in 'input' mode
EDITOR CORRUPT	Editor object program corrupt

Note

In onepass mode, all errors are irrecoverable. Of the above, 'ASL EMPTY' and 'EDITOR CORRUPT' can occur at any time, the others only in onepass mode.

Appendix 6

format	meaning	page
A '....'	after	I-7
B '....'	before	I-7
C '....'....'	change .... to	II-3
D '....'	delete	I-7
F '....'	find	I-6
I '....'	insert	I-7
N '....'	new line	II-3
U '....'	up to .... (delete)	I-7
πU '....'	multi-line 'U'	II-4
M (±) n	move ±n lines	I-9
P (+) n	print n lines on console	I-9
K (+) n	kill n lines	I-9
S (±) n	step ±n characters	II-3
X (+) n	exit n levels of bracketting	II-3
πNL (+) n	insert n newline characters	II-4
! (+) n	do the last command n more times	II-4
T	go to top of file	I-11
L	FP -> left margin	I-11
R	FP -> right margin	I-11
πLD	left delete	II-3
πRD	right delete	II-4
πCLOSE	close output file	I-11
πLISTCLOSE	'πCLOSE' + listing	II-4

(over)

format	meaning	page
subsidiary input		
::<newline>	enter/leave input mode	I-12
πREWIND	top of secondary input file	II-9
πGET (+) n	insert n lines	II-9
πSKIP (+) n	skip over n lines	II-8
πGETTO '....'	get up to and including	II-8
πSKIPTO '....'	skip up to and including	II-8
Onepass Mode		
πONEPASS	enter onepass mode	III-2
πLINE (+) n	move to line n	III-2
πAT (+) n	'πLINE' + selective listing	III-2
πNOLIST	suppress selective listing	III-2