27
5-12-77

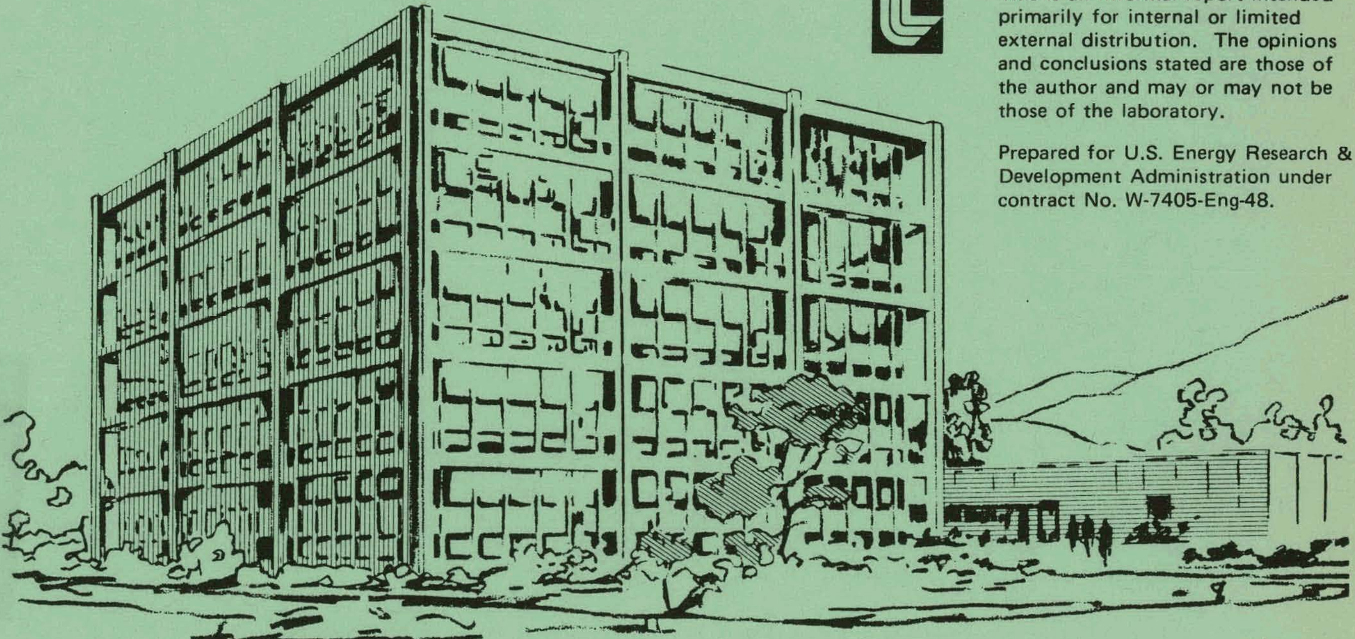
UCID-17372

Lawrence Livermore Laboratory

DATA DIRECTOR EDITOR USER'S MANUAL

P. McGoldrick

March 21, 1977



Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CONTENTS

DATA DIRECTOR EDITOR USER'S MANUAL

ABSTRACT

EDIT allows the user to edit or create text, either interactively or in the batch mode. The text may be a source program, program data, or documents. Special provisions permit the creation or editing of APT source-language programs. EDIT processes ASCII files from any accessible Data Director library and outputs files to any such library as specified by the user.

INTRODUCTION

The user is allowed to examine, change, or delete characters according to context or line reference numbers. EDIT may be operated from a teletypewriter, CRT, or batch input file. When using a terminal device, lines of text may be edited interactively. The entire file of text is created or read into a disk file called the "working file." After changes have been made to the working file, the file can be written to a specified output file.

All of the functions presented in this manual may not be implemented in the current edition of the editor. When in doubt, refer to the section "Unimplemented Functions" on page 25. Unimplemented functions are included in this manual for completeness and to ensure that no change is made to the editor that would disallow their implementation.

USE OF EDIT

To start EDIT, type "EDIT" on a user console and the message

```
EDIT Rmm/dd/yy
```

will be printed on the terminal, where mm/dd/yy is the revision date of the editor. In the interactive mode, EDIT responds with a period (.) when the editor is ready to accept a command. The user then types the desired command, terminating it with a carriage return. A carriage return is used to terminate any line typed by the user. When execution of a command is completed, the editor prints a period (.) on the terminal in expectation of another command.

More than one EDIT command may be input per line by typing a linefeed between each command. The linefeed is used as a logical end of command terminator and is echoed to the user as an exclaimation mark (!). The editor will process each command on the command list and prompt for more input when it has completed the list or has detected any error.

EDITOR ORGANIZATION

The basic operations of the editor are to:

- Create a working file by opening an existing file or creating an empty one.
- Modify the text in the working file.
- Save the contents of the working file by overwriting the currently open file or writing a new file.

The EDIT commands control these operations by context or line numbers. EDIT processes ASCII files containing variable length logical records, not to exceed 130 characters per line. These files are most often Filer files and can exist in any of the Data Director's directories.

Text Identification

Text in the working file or in an input file may be referred to by line number or by a character string within a line of text. Line numbers are consecutive numbers identifying the lines of the working file. They are determined by the order in which lines are read from the input file and are printed whenever lines of text in the working file are printed. For example, line numbers of a 300-line input file run from 1 to 300.

Lines of the text retain their original line numbers regardless of line insertions or deletions, except when the movement of lines of text or the insertion of text causes the current line number to be greater than its following line number. When this occurs, all following line numbers will be renumbered in 0.01 increments starting with the current line number, until the current line number is less than its following line number or until the end of the working file is reached. A decimal point followed by one or two decimal digits identifies inserted lines. Inserted lines may be numbered from 0.01 to 0.99. The line numbers are not written into the output file. For example, we begin with

```
100 line 100
101 line 101
102 line 102
103 line 103
104 line 104
```

If we delete line 101 and add two lines between line 102 and 103, we have

```
100 line 100
102 line 102
102.01 line 102.01  {inserted line}
102.02 line 102.02  {inserted line}
103 line 103
104 line 104
```

Lines or blocks of text in the working file may be identified by character strings (often called patterns) within lines of text. For example, a whole block of source text can be identified by the source labels that start in column one.

The current line pointer is a pointer that EDIT keeps to mark the "current line" in the working file. This pointer is valuable because the user can tell the editor to work on the current line rather than specifying a particular line number. The current line is defined by the execution of certain EDIT commands. The position of the current line pointer is noted in each command description, but as a rule of thumb, the current line is the last line operated on by the last edit command that was executed.

EXPLANATION OF EDIT COMMANDS

This section is a primer for those users who have had little or no experience with editors. A subset of EDIT's commands is discussed along with their use. A new user should find this section of greatest benefit if he follows it at an EDIT terminal, trying the commands as he reads. Most of the examples in the following sections will describe APT source programs, but the same principles hold for any text.

Start EDIT as described in the previous section, obtaining the start message

```
        EDIT Rmm/dd/yy
```

on the terminal along with the period (.) prompt on the next line. The editor is now ready to be commanded.

Creating a File

The user types the following:

```
User:  EDIT                 {User starts EDIT}
EDIT:  EDIT Rmm/dd/yy
User:  .C MYFILE            {Period typed by EDIT, creates MYFILE}
User:  .BL 1                {Insert text before line 1}
User:  *PARTNO LODGE & SHIPLEY  OUTER CONTOUR L-463 {EDIT types *}
User:  *UNITS/INCHES
User:  *(CR)                {User types only carriage return}
User:  .END                 {EDIT types period}
EDIT:  ALL DONE             {EDIT terminates, saving inserted lines}
```

The user starts the editor by typing EDIT on the first line. The user receives back the revision date of the editor and a period (.) to indicate that the editor is ready for a user command. The user then types the rest of the third line, which commands the editor to create a working file for MYFILE. When the editor completes the request, it again types a period for another command. The user then commands EDIT to start inserting lines before line 1 (because there are as yet no lines in the file).

EDIT enters the text-insertion mode and indicates the action by typing a "*" instead of a period (.) when it is available for more input. The next two lines the user types go directly into the working file. When the user finishes entering his two-line program, he types only a carriage-return in response to the editor's request for more text input. The editor interprets the solitary carriage-return as a request to exit the text-insertion mode and to return to the command mode.

The user then types END, which tells EDIT to terminate. EDIT copies the contents of the working file into a file named MYFILE, destroys the working file, and terminates. EDIT's termination is noted by the last line.

Editing an Existing File

Often a user wishes to EDIT an existing file. This includes correcting, updating, or modifing part programs or master decks. To do this, the user types the following:

```
User:  EDIT                {User requests editor}
EDIT:  Rmm/dd/yy
User:  .O MYFILE           {Open MYFILE}
EDIT:  2 LINES             {EDIT tells how many lines are in file}
EDIT:  .                   {EDIT prompts for next command}
```

The user starts the editor as before. But on the third line, the user opens the file MYFILE, causing MYFILE to be read into a working file. The editor responded with the number of lines in the file and a period requesting another command.

Listing the Working File

To print the contents of the working file on your terminal, use the list all command, LA:

```
User:  .LA 1                {List all lines, starting with line 1}
EDIT:  1 PARTNO LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT:  2 UNITS/INCHES
EDIT:  .
```

Inserting Lines

To insert lines of text, use command BL (before line) or AL (after line). In the following example, the user has already used BL to create the original two lines of text, and now uses AL to add two more lines to the end of the file.

```
User:   .AL 2                {After line 2}
EDIT:   *PPOUTPUT/DATADIR
EDIT:   *INTOL/.00005
EDIT:   *FINI
EDIT:   *(CR)                {Single carriage return}
EDIT:   .
```

The lines between AL 2 and the carriage return are now in the working file.

```
User:   .LA 1
EDIT:   1 PARTNO LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT:   2 UNITS/INCHES
EDIT:   3 PPOUTPUT/DATADIR
EDIT:   4 INTOL/.00005
EDIT:   5 FINI
EDIT:   .
```

To list only the middle two lines, type: LA 2,3. The command can be read to say, "list all lines in the range line 2 through line 3." After the LA command the current line is the last line printed.

Deleting a Line

To delete a line of text, type the line number of the line to be deleted, followed immediately by a carriage return. This command does not change the line numbers of any other lines in the working file. In the following example, the user deletes line 4 and lists the working file.

```
User: .4(CR)
User: .LA 1               {Remember EDIT typed the .}
EDIT: 1 PARTNO LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT: 2 UNITS/INCHES
EDIT: 3 PPOUTPUT/DATADIR
EDIT: 5 FINI
EDIT: .
```

Inserting a Line

To insert a new line at the end of the working file and list the new line, the user types:

```
User: .4 MACHN/LODSHP
User: .LF 4
EDIT: 4 MACHN/LODSHP
EDIT: .
```

After the user inserts the new desired line by typing the desired line number followed by the desired line, the user asks the editor to list line 4 with the LF 4 command. LF 4 can be read as, "list the first line in the range line 4 to the end of the file."

To insert a line between two existing lines, take the line number of the line after which the insertion should take place and add 0.01 to it. If there is no line with this number, then use it as above to insert the line. The line will be inserted in the appropriate place in the file:

```
User: .3.01 MACHIN/GRAPHICS,XYPLAN,-1,11,-6,6
User: .LA 1
EDIT: 1 PARTNO LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT: 2 UNITS/INCHES
EDIT: 3 PPOUTPUT/DATADIR
EDIT: 3.01 MACHIN/GRAPHICS,XYPLAN,-1,11,-6,6
EDIT: 4 MACHN/LODSHP
EDIT: 5 FINI
EDIT: .
```

In general, any line number can be used to insert a line as long as the new line number is between the existing two line numbers and has two or less decimal places (e.g., in the above example: 3.1, 3.5, 3.81 or 3.99 would have worked equally well as the new line number).

Any number of inserted lines may be placed between two line numbers until all the numbers are used. When this occurs, use AL or BL. With AL or BL, when the line number of the last inserted line is greater than the line number following, all following numbers will be renumbered in 0.01 increments until the current line number is less than the following line number.

```
User: .AL 3
User: *OUTTOL/.00005
User: *(CR)
User: .LA 1
EDIT: 1 PARTNO/LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT: 2 UNITS/INCHES
EDIT: 3 PPOUTPUT/DATADIR
EDIT: 3.01 OUTTOL/.00005
EDIT: 3.02 MACHIN/GRAPHICS,XYPLAN,-1,11,-6,6
EDIT: 4 MACHN/LODSHP
EDIT: 5 FINI
EDIT: .
```

Replacing a Line

Instead of deleting and reinserting a line, the user may simply replace a line directly.

```
User: .2 UNITS/MM
User: .LA 1
EDIT: 1 PARTNO/LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT: 2 UNITS/MM
EDIT: 3 PPOUTPUT/DATADIR
EDIT: 3.01 OUTTOL/.00005
EDIT: 3.02 MACHIN/GRAPHICS,XYPLAN,-1,11,-6,6
EDIT: 4 MACHN/LODSHP
EDIT: 5 FINI
EDIT: .
```

Pattern and Symbol Substitution

To substitute one string of characters for another in a line, the user may use the RFP or RAP command.

```
User: .RFP 4/HN/HIN/
User: .LF                {List current line, which is line 4}
EDIT: 4 MACHIN/LODSHP
EDIT: .
```

The user notices that MACHN in line 4 is misspelled, so the user requests the editor to replace the pattern, HN, with pattern, HIN, in the first line containing that pattern, starting with line 4. The character slash (/) is a delimiter that tells the editor the boundaries of the pattern and replacement. Either a slash (/), question mark (?), or CRTL-P (echoed $) character may be used as delimiters. There are three other delimiters that can be used; but they put further constraints on the pattern to be replaced. These delimiters are semicolon (;), quote ("), and CTRL-S (echoed $). When any of these latter three delimiters are used to delimit a pattern, the editor will only replace the pattern if there are nonnumeric, nonalphabetic characters to the left and right of the pattern in the working file. These especially constrained patterns are called symbols.

HN is a symbol	HN is a pattern
HN	HNI
HN/R4	MACHN
,HN.4	,HN4
(HN)	(AHN)

Note that, to list the changed line, the user uses LF with no arguments.

If the user wishes to change all occurrences of the pattern, HN, with pattern, HIN, in the whole file, the user uses RAP 1/HN/HIN/ instead of RFP. In the simple case above, both commands produce the same results because there is only one occurrence of pattern, HN. The results would have been drastically different if the user had used RAP 1/N/IN/ instead.

Moving and Copying Lines

To move a line or group of lines to another location in the working file, use the MA (move after) command:

```
User: .MA 4,3.01,3.01
User: .LA 1
EDIT: 1 PARTNO/LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT: 2 UNITS/MM
EDIT: 3 PPOUTPUT/DATADIR
EDIT: 3.02 MACHIN/GRAPHICS,XYPLAN,-1,11,-6,6
EDIT: 4 MACHIN/LODSHP
EDIT: 4.01 OUTTOL/.00005
EDIT: 5 FINI
EDIT: .
```

In the first line of the example above, the user tells the editor to move lines 3.01 through 3.01 after line 4. With the MA command, line 3.01 is deleted from the working file and is inserted after line 4.

Lines can be copied rather than moved with the CA command. CA works like MA, except that the lines being copied are not deleted from the working file.

```
User: .CA 4,4.01,4.01
User: .RFP /OUT/IN/
User: .LA 1
EDIT: 1 PARTNO/LODGE & SHIPLEY OUTER CONTOUR L-463
EDIT: 2 UNITS/MM
EDIT: 3 PPOUTPUT/DATADIR
EDIT: 3.02 MACHIN/GRAPHICS,XYPLAN,-1,11,-6,6
EDIT: 4 MACHIN/LODSHP
EDIT: 4.01 INTOL/.00005
EDIT: 4.02 OUTOL/.00005
EDIT: 5 FINI
EDIT: .
```

The user copies lines 4.01 through 4.01 after line 4. This places another OUTTOL line after line 4. Then, the user replaces pattern OUT with pattern IN, producing INTOL in the above example. Note that the current line for Move or Copy commands is set to the last line moved or copied.

Saving the Working File

To save the contents of the working file, it is necessary to copy the working file to an output file. The command NF writes the entire working file over the one named on the open, create, or NF command line (see page 11).

The NF command is most often used when one desires to preserve the original file that was opened in its original form or update the file at selected editing intervals. An automatic NF is done by the editor when the user types END if there is an active working file.

Security

A security level can be assigned to the working file when a file is opened, created, or NFed. The editor only labels files with a security level. No action is taken by the editor to either allow or deny access because of security.

To assign a security level, simply put .U, .P, .A,, or .S on the open, create, or NF line. The security levels are:

```
.U - Unclassified                          {default on create command}
.P - Protect as Restricted Data (PARD)
.A - Administative
.S - Protect as Secret Restricted Data (SRD)
```

When a security level is used, from that moment on, the level of the working file will be at the assigned level, as will any subsequent NFs, until the level is changed by another assignment command. As an example, on page 3 the user could have created an SRD file by typing:

```
C .S MYFILE
```

A file being opened already has a security level associated with it. If the file's security level is higher than the level on the open command line then the editor's security level will be set to the level of the file. For example, opening a PARD file:

```
USER: O .U MYFILE
EDIT: 8 LINES
EDIT: PARD
```

Notice that if the level of the file is greater than unclassified, the level is printed when the file is opened. If the user had typed O .S MYFILE, then the level would have been set to SRD. As a rule, the security level will be set to the higher of the level of the file and the level on the command line.

Security levels can be changed absolutely by the NF or END command. This is the only way declassification can be accomplished. If the security level on the NF or END command is defaulted, then the security level of the new file will be set at the level to which the editor was set when the file was opened or created. Thus, from a security standpoint, security assignments are best made with the open or create command.

File Properties

A file can have a property associated with it that serves to protect the file from inadvertant tampering. The properties are:

```
RW - Read/write, the file can be read or written.
R  - Read-only, the file can only be read.
```

Properties, like security, can only be made the same or more restrictive by the open command. A read-only file, opened with the command O RW. MYFILE, will result in all subsequent files remaining read-only. Read-only files can be made read/write with the NF or END command only.

DESCRIPTION OF EDIT COMMANDS

Commands to the editor may be classified into three basic types:

- File commands which move text between the input file, working file and the output file.

- Working file commands which operate on the text in the working file.

- Utility commands that control the environment in which the editor operates.

Command Format

The syntax and desription of all commands follow. Included with each description, if applicable, is the position of the current line and possible output from executing the command. A formal description of the editor syntax is given in in the Appendix.

An edit command consists of an operation code followed by a space and one or more optional operands. Descriptions of the commands are given with the optional operands in brackets. The following definitions are used:

PATTERN	Specifies a string of characters; any ASCII character is valid.
L	Specifies a line number.
m,n	Specifies decimal integers.
i,j	Specifies column numbers between 1 and 130.
,	(comma) delimits non-string operands.
RANGE	Specifies L,L/PATTERN/i,j {see Page 13}
REPLRANGE	Specifies L,L/PATTERN/PATTERN/i,j
/	When used in RANGE / can specify / or ? or CTRL-P to delimit PATTERNS; or ; or " or CTRL-S to delimit SYMBOLS.

File Commands

Command: C FILEID {create file}

Purpose: To create a working file for FILEID, where FILEID is the directory and filename of the file; the file's security and protection level are also specified.

Examples:

C MYFILE	{create file MYFILE}
C .PP:JOE	{create file JOE in directory .PP}
C .S PETE	{create file PETE at a security level of SRD}
C DELIA .U	{create file DELIA unclassified}

Security levels are:
- .U - Unclassified
- .P - Protect as restricted data (PARD)
- .A - Administrative
- .S - Secret restricted data (SRD)

Defaults: If the security level is omitted, the level is set to unclassified. If the file's property is defaulted then the file's property is set to read-write.

Discussion: Notice that the security level and directory:filename may appear in any order on the command line.

Command: O FILEID {open file}

Purpose: To open an existing file by copying it to a working file.

Examples:

O MYFILE	{Open MYFILE}
O .S PETE	{Open file PETE at a security level of SRD}

Defaults: If the security level is not specified, the level of the working file is set to the level of the file being opened. If the file's property is defaulted then the property will be set to that of the file. See pages 9 and 9 for more details on file security and properties.

Command: NF [FILEID] {new file}

Purpose: To replace or create FILEID with the text from the working file.

Examples:

```
   NF                 {replace currently open file}
   NF JOE
   NF .A JOE
   NF R. .S           {replace the current file with a new file
                       that is read-only and SRD}
```

Defaults and Discussion: If NF is used without a FILEID, the file that was created or opened by an O or C command will be created or replaced. NF is recommended as a way to provide a backup and to eliminate the need to copy a text file that you want to preserve in its original state (such as master decks). For example, the sequence

```
   O .MD:ICL435       {open master deck ICL435}
   make changes
   NF MYFILE .S       {put changed version into MYFILE at SRD level}
```

preserves the original master deck, ICL435, whereas

```
   O MYFILE
   make changes
   NF MYFILE1
   make changes
   NF MYFILE2
```

provides a backup.

Command: FF {forget file}

Purpose: To destroy the working file. This command allows the editor to abandon an incorrectly modified working file or a file that has been NFed.

Examples:

```
   FF       {forget the currently open file}
```

Discussion: The FF command may be followed by an O, C or END command.

Command: END [FILEID]

Purpose: To copy the working file over the currently open file or FILEID, destroy the working file, and terminate the editor.

Discussion: The same rules concerning security and property that govern NF cover END.

Working File Commands

Line Number Conventions for All EDIT Commands

All line numbers are inclusive: LA 1,10 means list all lines 1 through and including line 10. If the second line number is less than the first, then this second number is used as the number of lines to do. Thus, LA 50,3 means, starting at line 50, list 3 lines.

A line number can be a positive or negative offset from the current line:

LF -1 means list the first line before the current line.
LF +5 means list the fifth line after the current line.

There are three special characters that can be used as line numbers:

F = the first line of the working file.
L = the last line of the working file.
CL = the current line.

See EDIT Specifications, page 25, for limits on line numbers, increments, and offsets.

RANGE

All of the working-file commands perform some function on the working file. Most of these commands determine which lines are to be operated on by RANGE.

A user types:

```
LA 18,27
{and the editor lists}
18 CUTTER/C
19 PCEN=POINT/(A+J),0
20 C1=CIRCLE/CENTER,PCEN,RADIUS,R
21 L1=LINE/YSMALL,TANTO,C1,ATANGL,0
22 L2=LINE/PCEN,ATANGL,90
23 L3=LINE/PARLEL,YAX,XLARGE,A
24 L4=LINE/XLARGE,TANTO,C1,ATANGL,90
25 PT1=POINT/(A+J+R+.5+C/2),0
26 LPT1=LINE/PT1,ATANGL,90
27 PRINT/3,ALL
```

The LA command lists all lines in the RANGE, lines 18 through 27.

Often a user is more selective:

```
LA 18,27/C1/
{and the editor lists}
20 C1=CIRCLE/CENTER,PCEN,RADIUS,R
21 L1=LINE/YSMALL,TANTO,C1,ATANGL,0
24 L4=LINE/XLARGE,TANTO,C1,ATANGL,90
```

This LA command lists all lines between 18 and 27 that contain the PATTERN, C1, anywhere in the line.

Sometimes a user is highly selective:

```
LA 18,27/C1/1,2
{and the editor listed}
20 C1=CIRCLE/CENTER,PCEN,RADIUS,R
```

This LA command lists all lines between line 18 and 27 that contain the PATTERN, C1, starting in columns 1 through 2.

The defaults for RANGE are:

```
CL,L/any PATTERN/1,130
```

where: CL = the current line, L = the last line of the file, and

/any PATTERN/ = all lines.

Thus, the command LA 18,27/C1/,2 would produce the same results as:

```
LA 18,27/C1/1,2
```

or the sequence

```
LA 18,1      {position current line to line 18}
LA ,27/C1/1,2
```

Anything in RANGE may be defaulted with the exception that, if you drop out the /PATTERN/, you must drop out the column numbers.

First Line or All Lines in RANGE

Edit commands using RANGE either operate on the first line found in the RANGE or all the lines found in the RANGE:

```
LF RANGE means list the first line in RANGE.
LA RANGE means list all the lines in RANGE.

LF 18      {list the first line that exists in the RANGE, line
            18 to the end of file}
LF 1/A=/   {list first line starting from line 1 that contains A=}
LA 1/A=/   {list all the lines starting from line 1 that contain A=}
```

Command: L [text]   {line}

Purpose: To insert, replace, or delete a line by line number.

Examples:

```
5 this will be line 5 in the working file.
2.5 this may go between line lines 2 and 3.
7(CR)  {deletes line 7}
```

Current line: Set to the line inserted or replaced. If the line was deleted, then the current line is set to the line after the deleted line.

Discussion: See the earlier sections on inserting, replacing, and deleting a line.

Command: AL RANGE   {after line}

Purpose: To insert text after first line found in RANGE.

Examples:

```
AL 5
AL F/PARTNO/,1        {insert after PARTNO card}
AL L                  {insert after last line}
```

Current line: Set to last line inserted.

Discussion: Text inserted after the last line will be numbered with ascending integers, whereas text inserted between two lines normally will be numbered in increments of 0.01. The editor tries to use integer line numbers rather than decimal line numbers whenever possible.

Command: BL RANGE   {before line}

Purpose: To insert text before the first line in RANGE.

Examples:

```
BL                    {insert before the current line}
BL /FINI/             {insert before the FINI card}
BL F                  {insert before the first line in the file}
```

Current line: Set to the last line inserted.

Discussion: Text normally will be numbered in ascending increments of 0.01. The first line inserted will have the line number of the line before the first line in RANGE+0.01. The editor will attempt to use integer line numbers rather than decimal ones whenever possible.

Command: LF RANGE {list first}

Purpose: To list the first line in RANGE.

Examples:

```
   LF          {List the current line}
   LF 1        {List the first line that exists starting with line 1}
   LF 1,10     {List the first that exists in the RANGE, 1,10}
   LF /PPRINT/ {List the first line containing PPRINT after the
                current line}
```

Current Line: Set to the line listed if successful or to the last line searched if unsuccessful.

Command: LA RANGE {list all}

Purpose: To list all the lines in RANGE.

Examples:

```
   LA 1,10        {List all lines between 1 and 10 inclusive}
   LA 1/PARTNO/   {List all PARTNO cards}
   LA             {List all the file from the current line}
```

Current line: Set to the last line searched.

Command: P RANGE {page}

Purpose: To display a page of text starting at the first line in RANGE (a CRT page contains about 23 lines).

Examples:

```
   P 1        {display a page starting at line 1}
   P /PPRINT/ {display a page starting at a line containing PPRINT}
```

Current line: Set to last line displayed.

Command: Q RANGE {redisplay page}

Purpose: Redisplay the last page that was displayed by the P command.

Current line: Set to the last line displayed.

Discussion: This command, along with the P command, is only provided as a shorthand way of doing an LA startline,endline, where there is a page of lines between the startline and endline.

Command: DFL RANGE {delete first line}

Purpose: To delete the first line in RANGE.

Examples:

```
DFL F          {delete the first line}
DFL 1/GRAPHIC/ {delete the first GRAPHIC card}
DFL 17,33/C1/  {between lines 17 to 33 delete the first line
                containing C1}
DFL 100.1      {starting with line 100.1 delete the first line
                that exists}
```

Current line: Set to the line immediately following the deleted line or to the last line in the file.

Discussion: To delete a single line by line number, it is easier to use the alternate method (line number followed by carriage return, see page 5) rather than DFL. With DFL, if the line number that the user wishes to delete does not exist, then the first line that exists in RANGE, following where the nonexistent line should be, will be deleted.

Command: DAL RANGE {delete all lines}

Purpose: To delete all lines in RANGE.

Examples:

```
DAL 5,20        {delete all lines between 5 and 20 inclusive}
DAL 1/GRAPHICS/ {delete all lines containing GRAPHICS}
DAL             {delete all lines from the current line to the
                 end of the file}
```

Current line: Set to the last line searched, to the next line if the last line searched was deleted, or to the last line in the file.

Command: RFP REPLRANGE {replace first pattern}

Purpose: To replace the pattern(s) or symbols(s) in the first line in RANGE.

Examples:

```
RFP 27/XLARGE/YSMALL/ {starting with line 27 replace in the
                       first line containing XLARGE all
                       occurrences of XLARGE with YSMALL}
RFP 52,52/C1/C2/,1     {in line 52 replace the C1 that appears
                       starting in column 1 with C2}
```

Current line: Set to the line number in which that pattern was replaced if successful, or to the line number of last the line searched if unsuccessful.

Discussion: Those of us who are uncourageous, have little confidence in our eyesight or typing ability, and who wish to change a pattern in a specific line by line number, are advised to use the form:

```
   RFP L,L/PATTERN/PATTERN/ or
   RFP L,1/PATTERN/PATTERN/
```

Command: RAP REPLRANGE {replace all patterns}

Purpose: To replace all patterns or symbols in RANGE.

Examples:

```
   RAP 1/C1/C2/      {replace all C1 with C2 in the whole file}
   RAP 10,20;PT;L; {in lines 10 through 20 replace all symbols
                     PT with L}
```

Current line: Set to the last line searched.

Discussion: Notice in the second example above that ;PT; is a request for a symbol replacement. (See page 7.)

Command: CA L,RANGE [source FILEID] {copy after}

Purpose: To copy the lines in RANGE from the source FILEID and place them after the line, L, in the working file.

Examples:

```
   CA 5,1,10    {copy lines 1 through 10 from the working file
                 and put them after line 5 in the working file.}

   CA 3,1/EQU/10,10 ASMFILE {copy all the lines from ASMFILE
                             that contain EQU starting in column
                             10 and place the lines in the
                             working file after line 3}

   CA 12,F MAC1  {copy file MAC1 into the working file after
                  line 12}
```

Current line: Set to the last copied line.

Discussion: The line number (L) must not be defaulted. Lines from the source file are inserted after line L. The inserted lines are numbered as in command AL, starting with the line number L+0.01 and increasing in 0.01 increments. Renumbering occurs as with AL and BL.

CA allows the user to have macrofiles that can be inserted into the working file as desired (third example above). Notice from the first example above that if the FILEID is defaulted, then lines will be copied from the working file. Unfortunately, an anomaly occurs that makes L an illegal line number when it is used for the first line in RANGE with a file ID:

```
CA 2,L MYFILE     {Illegal}
CA 2,L            {legal}
```

Command: MA L,RANGE {move after}

Purpose: To move lines in range after line L.

Examples:

```
MA 7,10,35    {move lines 10 through 35 after line 7}
MA 10,9,1     {exchange line 9 and 10}
MA CL,-1,-1   {exchange current line with the line before it}
MA 5,F/C/1,1  {move all the lines beginning with C after line 5}
```

Current line: Set to the last line moved.

Discussion: The move takes place as follows:

```
1) TEMP = all lines in RANGE
2) DAL RANGE    {delete all lines in RANGE}
3) AL L         {after line L}
4) insert TEMP  {insert all lines in TEMP}
```

Utility Commands

Utility commands are commands that turn options on or off in the editor or are miscellaneous commands. Most of these commands do not, in themselves, cause anything to happen, but affect the way working file commands will operate. Many of these commands tell the editor what language the user's file is in, so that proper formatting or checking can be done.

Command: APTSYN ON/OFF {APT synonyms}

Purpose: To allow the user to turn on or off an option that will allow APT statements to be typed in a shorthand form that the editor will expand (see page 26). APTSYN ON keeps the synonyms on until the user types OFF or APTSYN OFF.

Examples:

```
User: .APTSYN ON
User: .1 C4=CI/XL,L1,YS,L2,RA,2
User: .LF
EDIT: 1 C4=CIRCLE/XLARGE,L1,YSMALL,L2,RADIUS,2
User: .RFP /XL/YS/
User: .LF
EDIT: 1 C4=CIRCLE/YSMALL,L1,YSMALL,L2,RADIUS,2
```

Discussion: The expansion of the synonyms takes place before the line is processed by the editor. Thus, synonyms for EDIT commands may exist.

Command: APTCK ON/OFF {APT check}

Purpose: To turn on or off the APT syntax checker that will verify the syntactical correctness of an inserted or modified line.

Discussion: This feature is under investigation at present and probably will not be defined well until later in 1977. This option is a separate program operated under the editor.

Command: ASM ON/OFF {assembly language}

Purpose: To decompress Interdata's compressed assembly language into a form that is more readily understood by the user.

Discussion: This option only affects lines that are displayed on a user terminal; each assembly language line remains in the form in which it was created or modified.

Command: BATCH FILEID

Purpose: To obtain the next editing commands from FILEID until an error or end-of-file occurs.

Discussion: Batching can be done to three levels (i.e., batch files can contain BATCH commands, but only three batch files can be active at one time).

Command: OFF

Purpose: To turn off all options: APTSYN, APTCK, ASM, and TC.

Command: TC ON/OFF {type changes}

Purpose: To turn on or off an option that automatically types, on the user's terminal, any line that is changed by the editor.

Examples:

```
User: .TC ON
User: .RAP 1/C1/C5/
EDIT: 20 C1=CIRCLE/CENTER,PCEN,RADIUS,R
EDIT: 20 C5=CIRCLE/CENTER,PCEN,RADIUS,R
EDIT: 21 L1=LINE/YSMALL,TANTO,C1,ATANGL,0
EDIT: 21 L1=LINE/YSMALL,TANTO,C5,ATANGL,0
EDIT: 24 L4=LINE/XLARGE,TANTO,C1,ATANGL,90
EDIT: 24 L4=LINE/XLARGE,TANTO,C5,ATANGL,90
EDIT: .

User: .APTSYN ON
User: .TC ON
User: .22 C1=LI/PC,AT,90
EDIT: 22 C1=LINE/PCEN,ATANGL,90
EDIT: .
```

Command: LOG ON/OFF

Purpose: To log to the user, all commands received by the editor.

Discussion: LOG ON is often used as the first command in a batch file and LOG OFF as the last. The user can then see the batch commands as they are accepted by the editor.

ERROR MESSAGES

The editor will try to let the user know where the editor was on the command line when the error was detected.

```
User: .LF 9.001
EDIT:          ↑
EDIT: ILLEGAL LINE NUMBER
```

Most often, the editor will point to a delimiter just after the offending information:

```
User: .O .BAD:JOE
EDIT:         ↑
EDIT: ILLEGAL DIRECTORY
```

An anomaly may occur when doing multiple commands delimited by the logical command terminator (!): the editor may point to the command after the one in which the error occurred:

```
User: .TC ON!LA ;X;!P F
EDIT:                ↑
EDIT: NONE FOUND
```

This means that the editor was ready to process the "P F" command but was unable to because no lines were found by the LA ;X; command. The editor always stops processing a multiple string of commands on any error.

ADVANCED EDITING

Often a user will use a subset of the EDIT commands that work nicely for him, but there may be some powerful things that the editor can do that may not be apparent from the previous descriptions. Also, most users (the author included) cannot visualize all the contexts in which a command can be used. This chapter then is a catch-all for any obscure command or command sequence.

Symbolic Line Numbers

By combining information from the earlier sections on replacing a line and on line number conventions, we obtain some handy line replacements:

```
User: .TC ON
User: .F THIS WILL REPLACE THE FIRST LINE IN THE FILE
EDIT: 1 THIS WAS THE FIRST LINE
EDIT: 1 THIS WILL REPLACE THE FIRST LINE IN THE FILE
User: .L WORKS FOR THE LAST TOO
EDIT: 789 THIS WAS THE LAST LINE
EDIT: 789 WORKS FOR THE LAST TOO
User: .CL CHANGE CURRENT LINE TOO
EDIT: 789 WORKS FOR THE LAST TOO
EDIT: 789 CHANGE CURRENT LINE TOO
User: .-1 INCREMENTS ALSO
EDIT: 788 THIS IS ONE LINE BEFORE THE LAST
EDIT: 788 INCREMENTS ALSO
EDIT: .
```

The Null Pattern

A null pattern (LF // for example) always evaluates to true if a character exists between the column limits. Thus:

```
User: LA F//81
EDIT: Lists all lines in the file that are longer than 80 characters.
User: .RA F///1,20
EDIT: Editor deletes columns 1 through 20 in all lines, and deletes
      those lines whose length is less than 20 columns.
```

The following can be used to insert an assembly-language label:

```
User: .LF
EDIT: 1  LHI R10,100
User: R //LABEL /,1   {replace first character with "LABEL "}
User: LF
EDIT: 1 LABEL LHI R15,100
```

Renumbering the Working File

A working file may be renumbered with even, ascending integers by the command MA F,F.

APPENDIX

EDIT Specifications

Listed below are the specifications for the current implementation of the editor.

Smallest line number	0	
Largest line number	32767.99	
Smallest offset	-127	{LF -127}
Largest offset	+127	
Smallest increment	1	{LA 500,1}
Largest increment	127	
Smallest column number	1	
Largest column number	130	

Unimplemented Functions

Command functions that are not currently implemented are listed below with their expected date of implementation.

Unimplemented function	Implementation date
APTSYN {APT synonyms}	June 1977
APTCK {APT syntax check} ...	Undecided

EDIT Shorthand Forms

The editor allows the user to enter some heavily used commands in a shorthand form. The result obtained from a shorthand form is identical to that obtained by the standard form.

Short form	Standard form
A	AL
B	BL
D	DFL
DA	DAL
M	MA
R	RFP
RA	RAP

APT Synonyms

Synonyms are normally formed from the first and second letter of the APT reserved word. Those synonyms that are formed from the first and *third* letter are noted below with an asterisk (*), while those that follow no simple rules are noted with a plus (+).

AA	..... ATANGL*	PR	..... PRINT
CI	..... CIRCLE	QA	..... QADRIC
CL	..... CLPRNT	RA	..... RADIUS
CU	..... CUTTER	RF	..... REFSYS*
CY	..... CYLNDR	RM	..... REMARK*
DA	..... DATADIR	RS	..... RESERV*
DN	..... DNTCUT	RI	..... RIGHT
EL	..... ELLIPS	RL	..... RLDSRF
FE	..... FEDRAT	RT	..... RTHETA
FR	..... FROM	SI	..... SINF
GC	..... GCONIC	SL	..... SLOPE
GB	..... GOBACK*	SM	..... SMALL
GD	..... GODLTA*	SH	..... SPHERE*
GF	..... GOFWD*	SL	..... SPLINE*
GL	..... GOLFT*	SQ	..... SQRTF
GR	..... GORGT*	ST	..... STOP
GT	..... GOTO*	TB	..... TABCYL*
HY	..... HYPERB	TA	..... TANTO
IC	..... INCHES*	TE	..... TERMAC
IX	..... INDEX+	TH	..... THETAR
IV	..... INDIRV+	TI	..... TITLES
IT	..... INTOF*	TL	..... TLLFT
IL	..... INTOL+	TR	..... TLRGT*
JU	..... JUMPTO	TC	..... TRACUT+
LA	..... LARGE	TS	..... TRANSL+
LC	..... LCONIC	TT	..... TRANTO+
LE	..... LEFT	TF	..... TRFORM*
LI	..... LINE	UN	..... UNITS
MN	..... MACHIN+	VE	..... VECTOR
MC	..... MACRO*	VT	..... VTLAX
MX	..... MATRIX+	XL	..... XLARGE
MI	..... MIRROR	XS	..... XSMALL
MU	..... MULTAX	XP	..... XYPLAN*
NP	..... NOPPOUT*	XR	..... XYROT*
NR	..... NORMAL*	YL	..... YLARGE
NP	..... NORMPS+	YS	..... YSMALL
NU	..... NUMPTS	YP	..... YZPLAN*
OB	..... OBTAIN	YR	..... YZROT*
OU	..... OUTTOL	ZL	..... ZLARGE
PA	..... PARLEL	ZS	..... ZSMALL
PE	..... PERPTO	ZU	..... ZSURF*
PL	..... PLANE	ZP	..... ZXPLAN*
PO	..... POINT	ZR	..... ZXROT*
PP	..... PPOUTPUT		

EDIT Error Messages

I/O and FILER errors:

```
FATAL I/O ERROR xxxx
I/O ERROR xxxx
FILE NOT FOUND
FILER ERROR 04
FILER ERROR 05
FILER ERROR 06
FILER ERROR 07
FILER ERROR 08
FILE IN USE
FILER ERROR 10
DIRECTORY FULL
FILE ALREADY EXISTS
FILER ERROR 13
FILE IS READ ONLY
```

For an explanation of I/O and FILER errors, contact P. R. McGoldrick about a Data Director User's Manual.

Syntax errors:

```
ILLEGAL LINE NUMBER
ILLEGAL COMMAND
ILLEGAL FILE ID
ILLEGAL FILENAME
ILLEGAL DIRECTORY
ILLEGAL SECURITY
ILLEGAL PROPERTY
ILLEGAL PATTERN DELIMITER
ILLEGAL COLUMN NUMBER
ILLEGAL RANGE
ILLEGAL OPTION
```

Miscellaneous errors:

```
NONE FOUND
LINE TRUNCATED
A CURRENT FILE ALREADY EXISTS
NO OPEN FILE
LINE NUMBER OVERFLOW
MULTIPLE BATCH LEVEL EXCEEDED
FILE IS ILLEGAL TYPE
```

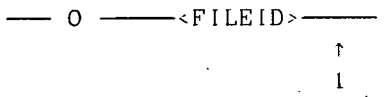
EDIT Syntax Diagrams

The EDIT syntax diagrams formally describe the legal EDIT commands. The diagrams may be of use when it is unclear from a command description whether a particular command form is legal or what results will be obtained. For example, a legal EDIT command is NF followed by a <FILEID>, see below. The results obtained after the evaluation of the <FILEID> is described below the diagram as item 2. If there was no <FILEID> after the NF command then the results obtained are described in item 1.

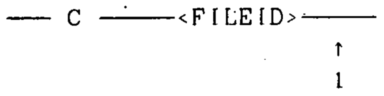
To find out what a <FILEID> consists of look at the left column in the diagrams until the definition of <FILEID> is located (page 36).

File Syntax

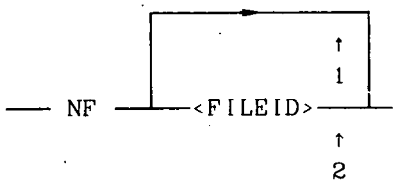
<COMMAND>



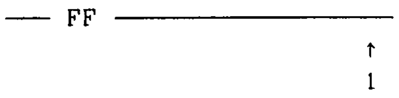
1) Open file <FILEID> by copying it to a working file.
   FILEID = <FILEID>



1) Create a working file for <FILEID>.
   FILEID = <FILEID>



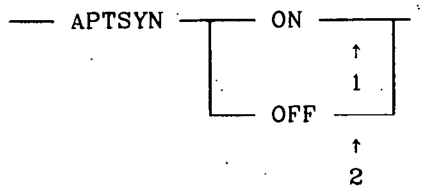
1) Write working file over FILEID.
2) Write working file over <FILEID>.
   FILEID = <FILEID>



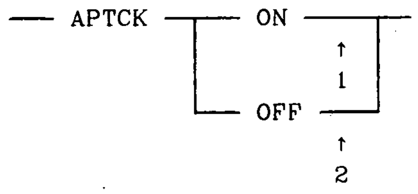
1) Destroy the working file.

Utility-Command Syntax

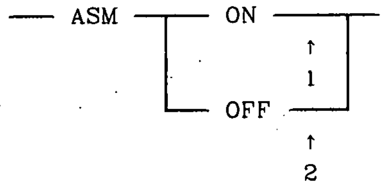
<COMMAND>



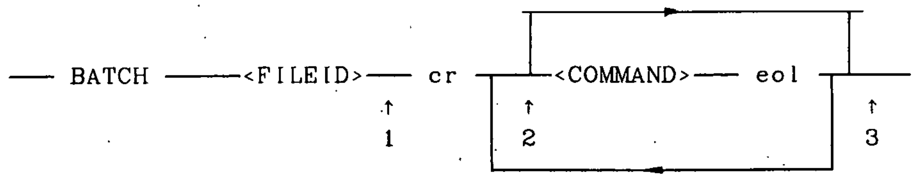
1) Turn on the APT synonyms.
2) Turn off the APT synonyms.



1) Turn on the APT syntax checker.
2) Turn off the APT syntax checker.

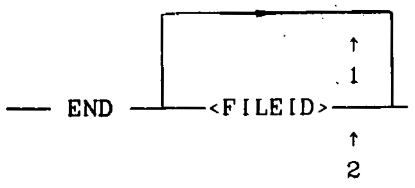


1) Turn on the assembly language formatting.
2) Turn off the assembly language formatting.

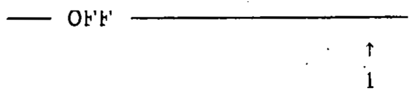


cr - carriage return or logical end of line terminator
eol -·end of linc

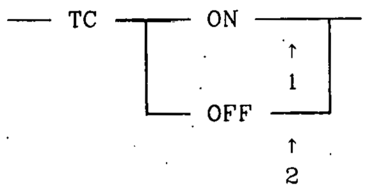
1) Open batch file, <FIL3.D>.
2) Obtain command from <FILEID> until error, the end of the file, or reaching an END command in the file.
3) Close <FILEID>.



1) Write out the working file over FILEID, destroy the working file, and terminate.
2) Write out the working file over <FILEID>, destroy the working file, and terminate.



1) Turn off all options: APTSYN, APTCK, ASM and TC.



1) Turn on "type changes" option.
2) Turn off "type changes" option.

File-Manipulation and Listing Commands

<COMMAND>

```
—— LF ———<RANGE>———
                  ↑
                  1
```

1) List the first line in <RANGE>.

```
—— LA ———<RANGE>———
                  ↑
                  1
```

1) List all lines in <RANGE>.

```
—— DFL ———<RANGE>———
                   ↑
                   1
```

1) Delete the first line in <RANGE>.

```
—— DAL ———<RANGE>———
                   ↑
                   1
```

1) Delete all lines in <RANGE>.

```
—— RFP ———<REPLACEMENT RANGE>———
                               ↑
                               1
```

1) In the first line in <REPLACEMENT RANGE>, replace all patterns with the REPLACEMENT PATTERN.

```
—— RAP ———<REPLACEMENT RANGE>———
                               ↑
                               1
```

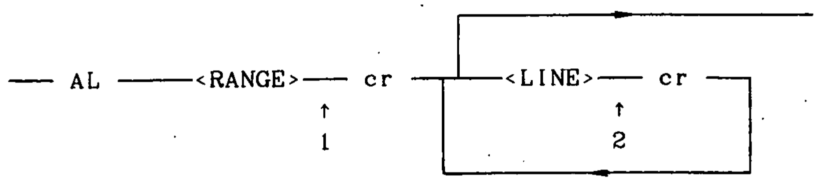
1) In all lines in <REPLACEMENT RANGE>, replace all patterns with the REPLACEMENT PATTERN.

```
—— P ———<RANGE>———
                 ↑
                 1
```

1) Display a page of lines starting at the first line in <RANGE>.
PS = first line in <RANGE>.

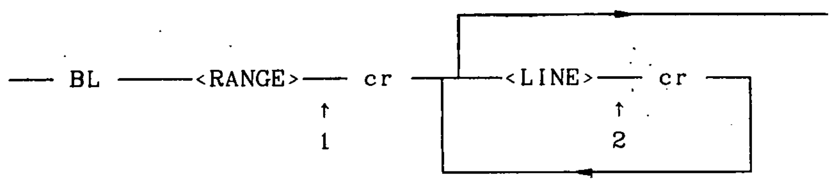
```
—— Q ———<RANGE>———
                 ↑
                 1
```

1) Display a page of lines starting from PS.
N.B. PS is set by by command P.

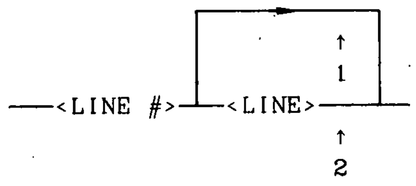


cr = carriage return or logical end of line.

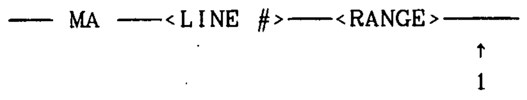
1) Current line = the first line in <RANGE>.
2) Insert line after the current line and set the current line to the inserted line.



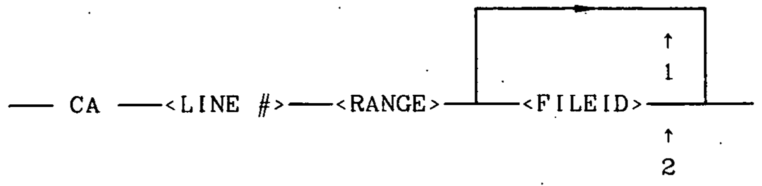
1) Current line = the line before first line in <RANGE>.
2) Insert line after the current line and set the current line to the inserted line.



1) Delete line <LINE #>.
2) Add or replace <LINE #> with <LINE>.



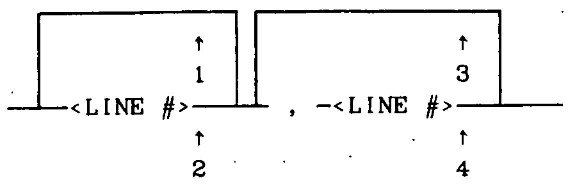
1) Move line(s) in <RANGE> after line <LINE #>.



1) Copy line(s) in <RANGE> from working file and insert them after <LINE #>.
2) Copy line(s) in <RANGE> from <FILEID> and insert them after <LINE #> in working file.

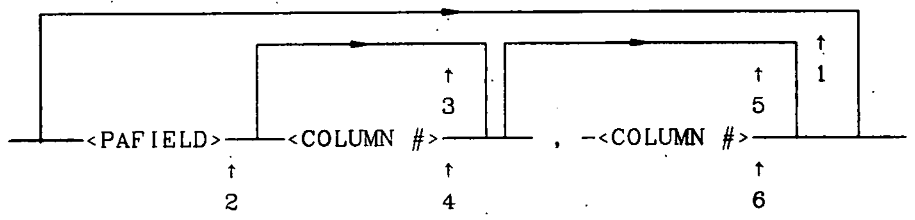
<RANGE>
——<EXPLICIT RANGE>——<ASSOCIATIVE RANGE>———

<EXPLICIT RANGE>



1) START LINE = CURRENT LINE.
2) START LINE = <LINE #>.
3) END LINE = Last line in file.
4) END LINE = <LINE #>.

<ASSOCIATIVE RANGE>

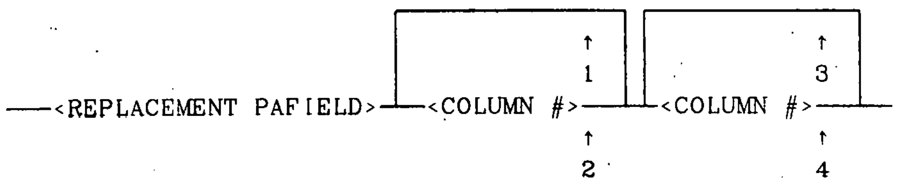


1) PAFIELD = null.
2) PAFIELD = <PAFIELD>.
3) START COLUMN = 1.
4) START COLUMN = <COLUMN #>.
5) END COLUMN = maximum column length.
6) END COLUMN = <COLUMN #>.

<REPLACEMENT RANGE>

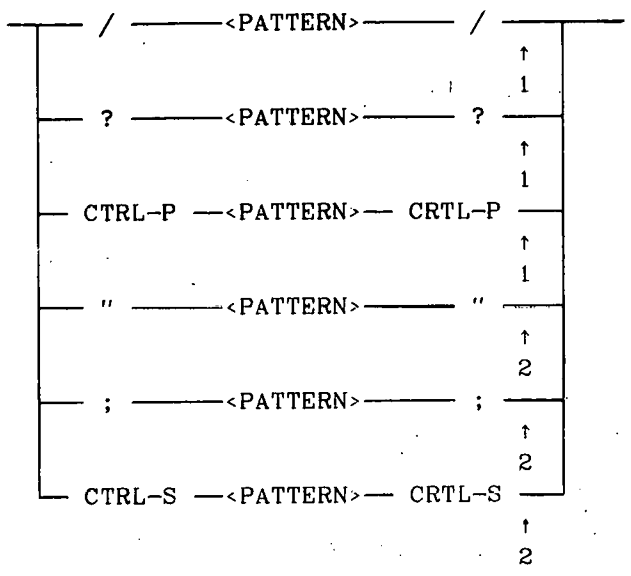
——<EXPLICIT RANGE>——<REPLACEMENT ASSOCIATIVE RANGE>———

<REPLACEMENT ASSOCIATIVE RANGE>



1) START COLUMN = 1.
2) START COLUMN = <COLUMN #>.
3) END COLUMN = maximum line length.
4) END COLUMN = <COLUMN #>.

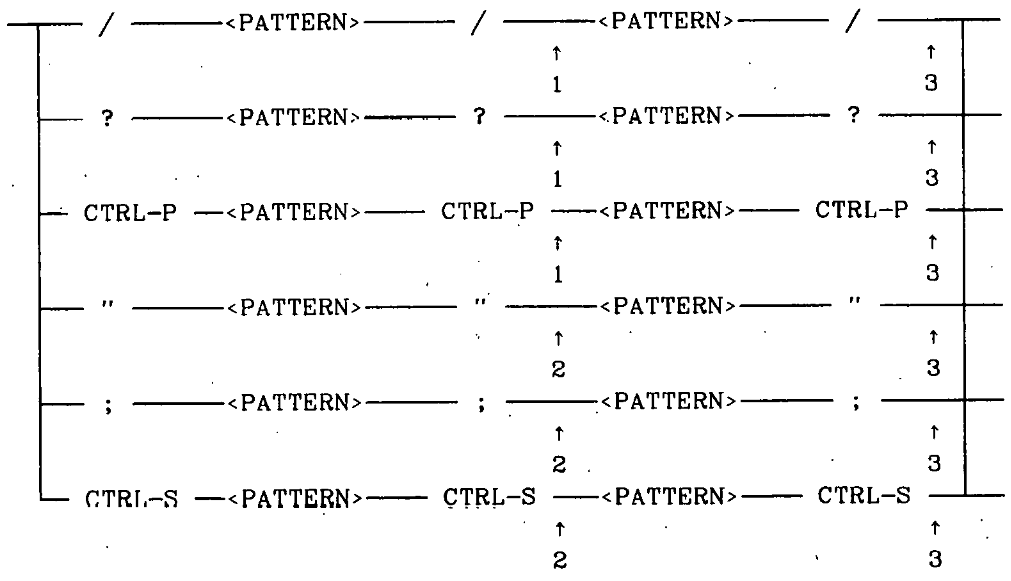
<PAFIELD>



1) PAFIELD = <PATTERN>; SYMBOL = FALSE.
2) PAFIELD = <PATTERN>; SYMBOL = TRUE.

(Symbol is a pattern that is constrained to have nonnumeric, nonalphabetic characters to the left and right of <PATTERN> in the file.)

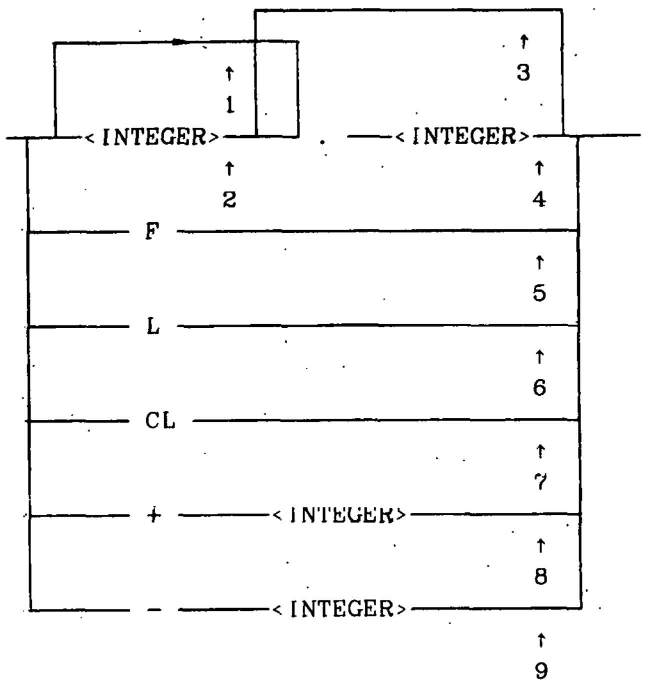
<REPLACEMENT PAFIELD>


/ <PATTERN> / 1 <PATTERN> / 3
? <PATTERN> ? 1 <PATTERN> ? 3
CTRL-P <PATTERN> CTRL-P 1 <PATTERN> CTRL-P 3
" <PATTERN> " 2 <PATTERN> " 3
; <PATTERN> ; 2 <PATTERN> ; 3
CTRL-S <PATTERN> CTRL-S 2 <PATTERN> CTRL-S 3

1) PAFIELD = <PATTERN>; SYMBOL = FALSE.
2) PAFIELD = <PATTERN>; SYMBOL = TRUE.
3) REPLACEMENT PATTERN = <PATTERN>.

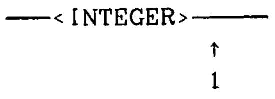
<PATTERN> = <any ASCII character>*
Here, <PATTERN> is equal to zero or more ASCII characters.

<LINE #>



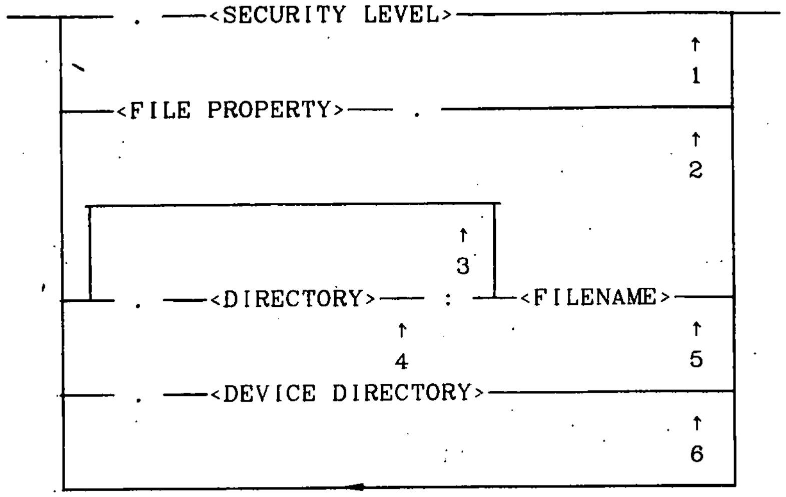
1) Set integer part of <LINE #> to 0.
2) Set integer part of <LINE #> to <INTEGER>.
3) Set decimal part of <LINE #> to 0.
4) Set decimal part of <LINE #> to <INTEGER>.
5) Set <LINE #> to first line in file.
6) Set <LINE #> to last line in file.
7) Set <LINE #> to current line.
8) Set <LINE #> to integer lines after current line.
9) Set <LINE #> to integer lines before current line.

<COLUMN #>



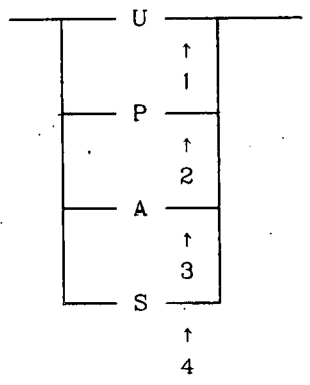
1) <INTEGER> is between 1 and maximum line length.

<FILEID>



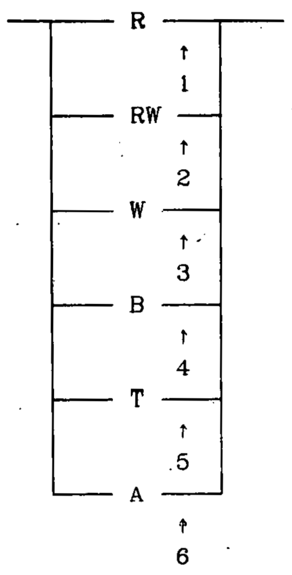
1) SECURITY LEVEL = <SECURITY LEVEL>.
2) FILE PROPERTY = <FILE PROPERTY>.
3) DIRECTORY = USER'S DIRECTORY
4) DIRECTORY = <DIRECTORY>.
5) FILENAME = <FILENAME>.
6) DIRECTORY = <DEVICE DIRECTORY>

<SECURITY LEVEL>



1) Unclassified.
2) Protect as Restricted Data (PARD).
3) Administrative.
4) Secret Restricted Data (SRD).

<FILE PROPERTY> (Subject to revision.)



1) Read-only access.
2) Read/Write access.
3) Write-only access {such as a printer}.
4) Binary file.
5) Tape punch file.
6) ASCII file.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy $; Microfiche $3.00

Page Range	Domestic Price	Page Range	Domestic Price
001–025	$ 3.50	326–350	10.00
026–050	4.00	351–375	10.50
051–075	4.50	376–400	10.75
076–100	5.00	401–425	11.00
101–125	5.50	426–450	11.75
126–150	6.00	451–475	12.00
151–175	6.75	476–500	12.50
176–200	7.50	501–525	12.75
201–225	7.75	526–550	13.00
226–250	8.00	551–575	13.50
251–275	9.00	576–600	13.75
276–300	9.25	601–up	*
301–325	9.75		

*Add $2.50 for each additional 100 page increment from 601 to 1,000 pages; add $4.50 for each additional 100 page increment over 1,000 pages.

PL/crs