DATA GENERAL
CORPORATION
Southboro,
Massachusetts 01772
(617) 485-9100

T Hodgman

PROGRAM

Paper Tape Editor

TAPES

Binary: 091-000001 - 06

ABSTRACT

The NOVA Editor is a routine allowing for editing of paper tape input to produce updated paper tape output. It is most commonly used to modify program source tapes in preparation for a new assembly. The Editor executes simple command strings requested from the teletype to modify text on either a character or a line level. The location of text is facilitated by means of string searches.

093-000018-01

TABLE OF CONTENTS

TABLE OF CONTENTS

EDITOR MANUAL

1. Introduction

The purpose of the Editor is to update an input file to produce an output file. This is accomplished by commands issued from the teletype. The commands enable the user to delete, insert, and search for strings of text. The command structure is versatile enough to allow changes at both the line and character levels. String searches provide a convenient means for both locating and replacing characters of text.

2. Logical Operation

The input file to the Editor is a continuous stream of characters. Logically, this stream of characters is considered to be segmented into pages. A page is a string of characters up to but not including a form feed character. A page is logically segmented into lines. A line is a string of characters up to and including a carriage return.

The editing process consists of the following steps:

1) Read a page from the input file into an edit buffer.

2) Modify this page based on Editor commands issued from the teletype.

3) Output this modified page to the output file.

An implicit character pointer (CP) is maintained by the Editor and is used to locate the object string for text modifications. This pointer should be thought of as residing between two characters. For example, the CP might be positioned as shown below:

```
LABØ
  ↑
  CP
```

If characters are inserted, they will be placed at the position of CP. Insertion of a "1", for example, would give the following string:

```
LAB1Ø:
    ↑
    CP
```

where it is noted that CP now resides after the inserted character.

3. Commands

3.1 Notation

The following notation will be used in this manual to describe the format of Editor commands and the description of text strings.

A space will be explicitly represented by the symbol "␣", *e.g.*

TWO␣WORDS

A carriage return will be represented by the symbol " ↓ ", *e.g.*

LINE01 ↓ LINE02 ↓

Control codes will be indicated by an up arrow followed by the symbol which would have printed if CTRL had not been depressed, *e.g.*

↑G is the BEL code (ØØ7)

The escape code (Ø33) serves a special function to the Editor and has no non-control counterpart. It will be represented by "$" (in operation, the Editor will transmit "$" to the teletype whenever escape is input).

3.2 Format

3.2.1 Single Command

A single command has the following format:

arg code string1$string2$

Each command has a one or two letter *code* which identifies it. These codes will be discussed in detail in Section 3.3.

Some commands may optionally be preceded by an argument, *arg*. Arguments are signed integers in the range:

$$\emptyset \leq |arg| \leq 2\emptyset47$$

Arguments are taken as decimal numbers unless preceded by a quotation mark. A quotation mark causes the number to be interpreted as octal. For example, the following commands are equivalent:

66M and "1Ø2M

Certain commands require one or two *strings* of characters as operands. If a string is given, it must be terminated by an escape character.

3.2.2 Command String

The Editor executes a command string as opposed to individual commands. The format of a command string is:

$$cmnd_1\ cmnd_2 \ldots cmnd_n\$\$$$

$Cmnd_i$ represents a command as described in Section 3.2.1. Command strings are terminated by two consecutive escape codes. If $cmnd_i$ ends with an escape code, only one escape is needed to terminate the command string. The Editor will not begin execution of any command within the string until the command string is terminated. For example,

-13B is not executed but -13B$$ is executed.

Carriage returns are ignored in the command string and may therefore be used to continue a command string on a new line.

3.3 Command Descriptions

3.3.1 Read (yank) a Page

To read a new page into memory, the following command is given:

*arg*Y

The argument, if used, is ignored. The next page of the input file is read into the edit buffer. The previous contents of the edit buffer are destroyed. After execution of the command, CP resides before the first character of the new page.

3.3.2 Append a Page

To append a page to the edit buffer the following command is given:

*arg*A

The argument, if used, is ignored. The next page of the input file is appended to the end of the present edit buffer. After execution of the command, CP resides before the first character of the appended page.

3.3.3 Output (punch) the Buffer

To output contents of the edit buffer, the following command is issued:

*arg*P or *arg*PW

The effect of this command is summarized below.

argument	effect
$\emptyset$ or none	output the entire edit buffer to the output file
$n > \emptyset$	output n lines, starting at CP; in no case output past the end of buffer
$n < \emptyset$	illegal

If P instead of PW is used, a form feed character is punched as the final character. If PW is used, no form feed is punched. This command has no effect on CP. For example, assume the edit buffer contains the following:

LINE□1 ↓ LINE□2 ↓ LINE□3 ↓
↑
CP

The table below shows the affect of various output commands.

Command	Output
P$$	LINE␣1 ↓ LINE␣2 ↓ LINE␣3 ↓ $\overset{F}{F}$*
1PW$$	E␣2 ↓
4P$$	E␣2 ↓ LINE 3 ↓ $\overset{F}{F}$
2PW$$	E␣2 ↓ LINE 3 ↓

A common command sequence used when no modification to a page is necessary is

YP$$

3.3.4 Output Leader or Form Feed

To output form feed or leader, the following command is issued:

*arg*F

The effect of this command is summarized below.

argument	effect
Ø or none	output a form feed
$n > Ø$	output n inches of leader (nulls) if $n < 100$; output 100 inches of leader if $n \geq 100$.
$n < Ø$	illegal

This command has no effect on CP.

3.3.5 Output and Read

The output and read function described in 3.3.3 may be accomplished by the following command:

*arg*R

Its effect is summarized below.

argument	effect
Ø or none	perform the command PY
$n > Ø$	perform the command PY n times
$n < Ø$	illegal

---

*$\overset{F}{F}$ is used to represent a form feed character.

CP points to the beginning of the last page read into the edit buffer after execution is complete.

3.3.6 Examine (type out) the Buffer

To examine contents of the edit buffer on the teletype, the following command is issued:

*arg*T

The effect of this command is summarized below.

argument	effect
Ø or none	type the entire edit buffer out on the teletype
$n > Ø$	type out n lines, starting at CP; in no case type past the end of buffer
$n < Ø$	illegal

This command has no effect on CP.

3.3.7 Move CP to the Beginning of the Buffer

The following command positions CP to the beginning of the buffer:

*arg*B

If an argument is specified, it is ignored.

3.3.8 Move CP to End of Buffer

The following command positions CP to the end of the buffer:

*arg*Z

If an argument is specified, it is ignored.

3.3.9 Move CP

CP may be moved from its current position by means of the following command:

*arg*M

The effect of this command is summarized below:

argument	effect
Ø or none	no effect
$n > Ø$	move CP forward n character positions, but never past the end of the buffer
$n < Ø$	move CP back n character positions, but never past the beginning of the buffer

Assume the buffer looks as follows:

```
ADD□2,3, SZC ↓ LAB:   □STA□3, CNST ↓
                      ↑
                      CP
```

The M command can effect the following:

command	effect
3M	ADD□2,3,SZC ↓ LAB:□STA□3,CNST ↓ (CP ↑)
-2M	ADD□2,3,SZC ↓ LAB:□STA□3,CNST ↓ (CP ↑)
4ØM	ADD□2,3,SZC ↓ LAB:□STA□3,CNST ↓ (CP ↑)

3.3.1Ø Delete Characters

To delete characters from the buffer, the following command can be issued:

*arg*D

This command is treated exactly as the M command, but in addition all characters which are passed over in the process of moving CP are deleted.

3.3.11 Move CP to Beginning of a Line

To move CP to the beginning of a line, the following command can be given:

*arg*L

The effect of this command is summarized below.

argument	effect
$n > \emptyset$	move CP forward past n carriage returns
$\emptyset$ or none	move CP backward past a carriage return, then forward one character
$n < \emptyset$	move CP backward $\lvert n \rvert + 1$ carriage returns, then forward one character

This command always positions CP to the beginning of a line. The command ØL positions CP to the beginning of the line where CP resides.

To move CP to the beginning of a line after first positioning CP to the beginning of the buffer, the following can be given:

argJ

This command is equivalent to the command string

Barg-1L

If a Ø or negative argument is given, CP is merely positioned to the beginning of the buffer.

3.3.12 Delete Lines

To delete lines from the buffer, the following command can be issued:

argK

This command is treated exactly as L, but the lines passed over are deleted. The effect of this command is illustrated below.

Initial Buffer	Command String	New Buffer
LINE□1 ↓ LINE□2 ↓ ↑ CP	-1K$$	LINE□2 ↓ ↑ CP
LDA□2,3Ø ↓ COM□2,3,SZR ↓ ↑ CP	1K$$	LDA□2,3Ø ↓ COM□2,3 ↑ CP
LDA□2,3Ø ↓ COM□2,3,SZR ↓ ↑ CP	ØK$$	LDA□2,3Ø ↓ ,SZR ↓ ↑ CP

3.3.13 Insert Characters

To insert characters into the buffer, the following command can be given:

I string $

The *string* is inserted into the buffer at the position of CP. After insertion, CP is positioned to the right of the last character inserted.

An alternate form for inserting one character is:

arg I

The argument is masked to seven bits and inserted.

3.3.14 String Searches

The Editor will perform a search of the buffer for a specified string if the following command is issued:

S string $

The Editor searches forward for *string* from the current CP position. If found, CP will be positioned after the last character of the first occurrence of *string*. If the end of the buffer is reached without finding *string*, a message will be printed indicating this (§5). CP will be repositioned to the start of the edit buffer in this case. Some examples using the search command are:

Before	Command String	After
LINE□1↓ LINE□2↓ ↑ CP	SLINE$$	LINE□1↓ LINE□2↓ ↑ CP
LINE□1↓ LINE□2↓ ↑ CP	SE□2$-2D$$	LINE□1↓ LINE↓ ↑ CP
LINE□1↓ LINE□2↓ ↑ CP	S2$-1DI1A$ 15I$$	LINE□1↓ LINE□1A↓↓ ↑ CP

A search for a string may be continued if not found in the current edit buffer by examining subsequent pages of the input file. This type of search is

called for by:

Nstring$

If the string is not found in the current buffer, the Editor executes a "PY" and continues its search at the beginning of the new buffer. This process is repeated until the string is found or the input file is exhausted. If the latter is the case, a message will be printed indicating this.

A search for a string may be continued if not found in the current edit buffer by examining subsequent pages of the input file but without punching the pages before that page in which the string is found. This type of search is called for by

Qstring$$

If *string* is not found in the current edit buffer, the Editor executes a "Y" and continues until the string is found or the input file is exhausted. If the latter is the case, a message will be printed indicating this fact. This command is identical to "N" except that the edit buffers unsuccessfully searched are not punched.

3.3.15 Change a String

A search for a string followed by a deletion of that string (if found) and an insertion of a second string may be effected by:

Cstring1$string2$

String1 is replaced by *string2* and CP is positioned to the first character after *string2*. If the end of the edit buffer is reached before *string1* is found, a message will be printed indicating this. Note that *Cstring1$$* merely deletes *string1*.

3.3.16 Copy Remainder of Input File

If the command "E" (end) is given, the current edit buffer will be outputted (unless empty) as well as the remainder of the input file. If an edit on an

input file is complete, "E" can be issued to copy the remainder of the input file into the output file.

3.4 Special Action Characters

Certain characters that appear in the command string cause special action by the Editor. These commands are single character, non-alphabetic codes.

3.4.1 Number of Characters in Buffer

If "=" is encountered in the command string, the decimal number of characters presently in the buffer will be printed on the teletype.

3.4.2 Number of Lines in Buffer

If ":" is encountered in the command string, the decimal number of lines in the buffer will be printed on the teletype.

3.4.3 Number of Current Line

If "." is encountered in the command string, the decimal line number in which CP resides will be printed on the teletype. For example,

15J.$$

will cause ØØ15 to be typed (since CP points to the beginning of line 15).

3.4.4 Insert with Tab

If a tab character ($\uparrow_t$I) is encountered in the command string, it will be interpreted exactly as if I$_b$a had been encountered, *ie*, tab is an implicit insert command.

3.4.5 Last Character Delete

Before a command string has been terminated, it is possible to delete characters from the string. If RUB OUT is depressed at any time during input of the command string, the last character entered in the string will be deleted. The Editor will also transmit back the character which was deleted.

Assume the following string has been typed:

ITEXT

If RUB OUT is now depressed twice and the command terminated, the teletype copy will appear as follows:

ITEXTTX$$

and only the characters TE will be inserted.

3.4.6 Erase or Stop Command Execution

The special character ↑C can be used in two ways. If encountered before a command string is terminated, the entire string is deleted, and the user may begin again. If encountered after a command string is terminated, but before a new one has been requested (§4), the Editor stops execution of the present command string, stops punching or typing, resets CP to the beginning of the edit buffer, and requests a new command string.

3.4.7 Reset for Input

If more than one input tape is used, the special character ↑T should be typed between tapes. This character performs the same function as ↑C and in addition stops the input device and clears the input buffer. A new tape may now be mounted in the input device.

4. Operating Procedure

4.1 General

An object tape of the Editor will be provided with the standard NOVA software package. This object tape must be loaded by the binary loader (§093-000003). Once loaded, control will be transferred to the Editor. The Editor will delete its edit and input buffers and request the user to assign his input and output files. The Editor first requests the output device by typing

TTO(1) OR PTP(2)?

The user should respond by depressing 1 for teletype output or 2 for high speed punch output. The input device is requested by:

TTI(1) OR PTP(2)?

Respond with 1 for teletype input or 2 for high speed reader input. If the user wishes to punch his output tapes with parity, he should respond with 1 to the next question:

PARITY OUT(1) OR NOT(2)?

Finally, if the input tapes are punched with parity, the user should respond with 1 to the last question:

PARITY IN(1) OR NOT(2)?

Once the I/O devices have been assigned, the Editor will request a command string by typing "*" on the teletype. The user may now enter a command string which will be executed upon termination. A new command string may be given after the Editor again types "*". This procedure is repeated until the update is complete.

If it is necessary to reassign the input/output files and delete the buffers, perform the following:

1. Press RESET
2. Enter ØØØØØ2 in the data switches
3. Press START

If it is necessary to force the Editor to request a command string but not to affect the I/O assignments or current edit buffer, perform the following:

1. Press RESET
2. Enter ØØØØØ3 in the data switches
3. Press START

4.2 Input

Three characters are ignored on input by the Editor. These characters are:

Character	Octal Value
null	ØØØ
line feed	Ø12
rub out	177

Although line feed is ignored on input, the Editor will provide a line feed after every carriage return sent to the output file.

If parity is checked on input, a message will be printed on the teletype if a character read fails parity. The message gives the line number in which the parity error occurred, and the character in error will be represented by a "\" character. If the line number is *n*, the line may be examined by:

*n*JIT$$

4.3 Editor Output to Teletype

4.3.1 Invisible Characters

Any invisible character which the Editor transmits to the teletype will be visibly represented as an up arrow followed by the printing character which corresponds to the invisible character. For example, if the user inserts a $\begin{matrix}C\\R\\L\end{matrix}$TG, the Editor will respond with ↑G to the teletype.

4.3.2 Tab and Form Feed

A switch in the Editor is examined to determine whether to simulate tabs with spaces. If tabs are simulated, the predefined tab positions occur at columns 1, 9, 17, 25, etc. This switch is initialized to simulate tabs. It may be complemented, causing the Editor to supply the actual tab character to the teletype, by typing a ↑P in a command string. Each occurrence of ↑P causes the switch to be complemented, thus ↑P ↑P has no effect.

4.4 Teletype Selected as Output File

A problem arises if the teletype is used both for command input and the output file. If the punch is left in the "ON" position while typing Editor commands, these characters are punched as part of the output. The Editor resolves this problem in the following way. Before transmitting "*" to indicate an Editor command is called for, the Editor will halt with the Carry light off. This signals the user to turn the teletype punch OFF. After doing this, the user can press CONTINUE, causing the Editor to now request a command. After a command string is terminated, the Editor determines whether output is to occur. If it is, it again halts, this time with the Carry light on. This signals the user to turn the punch ON and press CONTINUE, causing the output to be punched. This process is repeated between commands and output updates, thus avoiding the interference problem.

5. Error Detection

Various errors can occur during the editing process. An error message will be given to indicate the type of error. These messages are discussed below.

5.1 Unsuccessful String Search

If a string search is initiated and the Editor is unable to locate the string, the following will be printed:

STR NOT FOUND

The user should check his command to make sure the string he is searching for is specified correctly.

5.2 Parity Error on Input

If a parity error is detected while reading, the following will be printed:

PARITY ERROR IN LINE NUMBER *dddd*

Where *dddd* represents the line number in which the error occurred. When this line is examined, the character in error will have been replaced by "\".

5.3 Edit Buffer Capacity Exceeded

If a user attempts to append a page to a buffer that is full, the following message will be printed:

BUFFER IS FULL-CANNOT DO A

The user should output the present buffer to enable reading of additional pages.

During the reading of a page, the buffer capacity may be exceeded. The following message will be typed:

BUFFER IS FULL-Y OR A INPUT TERMINATED

This indicates a partial page has been read into (or appended to) the edit buffer. This buffer must be outputted, or sections deleted, before the rest of the page can be read.

Since the command string also requires storage, it is possible for the command string storage to approach overlap with the edit buffer. If this should occur, the following will be printed:

BUFFER CAPACITY EXCEEDED DURING COMMAND INPUT
COMMAND IS TERMINATED AND BEING EXECUTED.

The Editor begins execution of the command string as if it had received two escape characters. Upon completion of the execution, the user can continue his edit from where it was terminated.

5.4 Questionable Command

If at any point the Editor is unable to understand a command, it will print the following:

?? *remainder of questionable command string*

The portion of the command string which has not been executed should be examined to determine the incorrect command. The user may continue by correcting the command and starting from the point at which execution ceased.

Edit Affecting Commands (cont.)

Command	Form	Function
R	*n*R	Perform PY *n* times
S	S*string*$	Search for string
T	T *n*T	Type entire edit buffer Type *n* lines from current CP
Y	Y	Yank a page into the edit buffer
Z	Z	Position CP to end of edit buffer

Special Action Commands

Command	Function
=	Print number of characters in edit buffer
:	Print number of lines in edit buffer
.	Print line number where CP resides
↑I (tab)	Interpret as I_a^t. $_b$
RUB OUT	Erase last character
↑C	Erase command string or halt execution of command string
↑T	Reset input buffer and stop input device

APPENDIX A

Command Summary

Edit Affecting Commands

Command	Form	Function
A	A	Append a page to the edit buffer
B	B	Position CP to the beginning of the edit buffer
C	C*string1*$*string2*$	Change *string1* to *string2*
D	*n*D	Delete *n* characters starting at CP
F	F *n*F	Punch a form feed Punch *n* inches of leader ($n \leq 100$)
I	*n*I I*string*$	Mask *n* to seven bits and insert at CP Insert string starting at CP
J	*n*J	Jump *n* lines from beginning of edit buffer
K	*n*K	Delete (kill) *n* lines from current CP
L	*n*L	Position CP *n* lines from current CP
M	*n*M	Position (move) CP *n* characters from current CP
N	N*string*$	Search for *string*; if not found, punch, read, and continue search
P	P *n*P	Punch entire edit buffer followed by a form feed Punch *n* lines from current CP followed by a form feed
PW	PW *n*PW	Punch entire edit buffer Punch *n* lines from current CP