the digital group
po box 6528 denver, colorado 80206 (303) 777-7133

EDITOR/FORMATTER
VOLUME 1

296-007-A-25

Foreword

This is Volume 1 of a two volume set describing the Editor/Formatter program. This volume provides the users instructions detailing how to operate the program. Volume 2, the Technical Manual, contains the theory of operation and the complete program listing.

ADDENDUM

This addendum lists the modifications which have been incorporated into the program for operation on a Digital Group System.

1. The program is initialized to the upper and lower case terminal mode, rather than to the upper case only mode.

2. The SETO command does not modify the upper/lower case mode of the terminal or printer. The upper/lower case mode is affected only by the TERM command.

3. The program has been relocated to accommodate the Digital Group cassette operating system, and all I/O patches have been incorporated in the program tape. The program starting address is 020_000 (octal), and the checksum routine begins at 044-367. The correct checksum value is 312-122.

INTRODUCTION

This program consists of a general purpose text editor and a text formatter. The editor is designed for the generation of any type of text files, including general text such as letters and reports, or source code files for programs to be processed by an assembler or compiler. The editor contains all of the text manipulation capability necessary to allow generation of these files in an easy and efficient manner.

The text formatter is specifically designed for the generation of letters, manuscripts, reports, or any type of material where it is desired to generate output with a neat appearance. The formatter handles margin justification, titles, page numbers, and all of the other details required for generation of this type of material. The editor/formatter turns the computer into a powerful word processing system.

The program is written in 8080 assembly language and occupies over 5K of memory, exclusive of the I/O routines. Memory space for the file to be processed must also be available. The minimum recommended memory is 12K, although a smaller system could be used. Text material can use up memory rapidly. For example, a page of single spaced typed material requires over 3K of memory for storage. I am currently running the program with 16K of main memory, which allows 3 to 4 pages of material to be stored on line. Longer files, such as this documentation, are handled as multiple records on tape.

The peripherals supported by the program are an ASCII terminal, a printer, and a cassette tape interface. In my case, the terminal consists of a keyboard and a CRT display, the printer is a Selectric typewriter, and the cassette interface is a Dr. Suding type (i.e. Digital Group). All I/O requirements are handled through a jump table, and the I/O routines must be supplied by the user.

THE EDITOR

The editor operates on an ASCII file which is stored in memory. The active file length is limited only by the available memory size. The editor is line/string oriented, and does not require line numbers to be contained in the file. The editor has the capability to search for and locate any string of characters, irrespective of its location in the file or within the line. Lines can be added, deleted, replaced, modified, or printed with simple input commands. The editor allows two or

more files to be stored in memory concurrently, and has the capability to selectively edit each file or merge them together.

The character set utilized by the editor consists of the standard upper and lower case printing ASCII characters. In addition, the program will accept four control characters: carriage return, which is used as the end of line terminator; control T, for inputting tabs; control C, which will cause a break in an output listing; and control S, which is used as a shift key on an upper case only terminal.

All commands to the editor are input from the terminal. While the file may contain both upper and lower case characters, the commands must be input as upper case characters. Optional parameters associated with the command may be upper or lower case characters. Commands are terminated with a carriage return (C/R), which signals the editor to begin processing the command. The input is buffered, so that input errors may be corrected before the C/R is input. The buffer will accept a line up to 132 characters long, and will truncate longer input lines to the 132 character length.

The commands recognized by the editor may be classified into three general categories: Initialization, Edit, and Utility. All commands must be followed by a space and/or terminated by the C/R. Additional parameters associated with a command (numerical or string data) must be separated from the command by one or more spaces. The editor outputs a greater than symbol (>) as a prompt, indicating that it is waiting for a command.

<u>Initialization Commands</u> - The program uses several internal pointers which define the currently active file. When the program is first loaded, these pointers must be initialized to correspond to the file to be edited. This is accomplished by using one of the initialization commands. These commands request the file starting address, which is input in split octal from the keyboard. The address input routine tests each character for valid octal data as it is input. An input error while inputting the address will immediately generate an error message, requiring the command to be reentered.

The three initialization commands and their functions are:

NEWF Directs the editor to open a new file at the specified address and enter the input mode. Use this command to input a new file from the keyboard.

LOAD Directs the editor to load an existing file from tape. Use this command to edit a file which has been previously saved on tape. The load begins when a C/R is input following the requested address, which allows time to set up the tape recorder. The command may be aborted following the address input by entering a question mark just prior to the C/R input.

EDIT Directs the editor to access a file which is currently contained in memory. Use this command if you start the program with a file already loaded into memory, or to transfer between multiple files in memory. This command may be abbreviated E.

Edit Commands - The edit commands are used to display and/or edit lines within the file. All edit commands operate on, or with respect to, the current line. In most cases, the current line is defined as the last line displayed on the terminal. The program uses a line pointer which always contains the starting address of the current line. This address changes as different lines within the file are accessed.

In the following command descriptions, a string is defined as any sequence, of any length, of valid ASCII characters. The number symbol (#) denotes an optional number associated with the command. The number is input as a decimal integer, and unless otherwise noted, must be in the range of 0 to 65535. A default value of 1 is assumed if no number is input. Where output is generated by a command, the output may be directed to either the terminal or printer. Input from the keyboard will always be echoed to the terminal, even if the output device is set to the printer.

The edit commands and the functions they perform are:

A String	Append the string to the end of the current line and display the result.
BOTM	Set the current line pointer to the end of the file.
C %String1%String2	Find the first occurence of String1 in the current line and change it to String2. The two string lengths need not be equal, and the second string can be null (i.e., a C/R following the second delimiter). The delimiters (%) may be any printing character. The result of the change is displayed.

D #	Delete the current line (or # lines) from the file. The current line becomes the line following the last line deleted.
F String	Find and display the first line in the file which begins with the string. The search begins with the line following the current line and continues until a match is found or the end of file is reached. The found line becomes the current line.
I String	Insert the string as a new line following the current line. If no string is included, or if only a C/R is input as a command, the editor enters the continuous input mode. In this mode, multiple lines may be entered in a file by typing in each line followed by a C/R. Exit from the continuous input mode is accomplished by inputting a null line (C/R only). When the continuous input mode is entered, the message INPUT will be displayed. Upon exiting the input mode, the message EDIT will be displayed. No prompt is issued between multiple input lines, which indicates that the editor is in the input mode.
L String	Locate and display the first line in the file which contains the string anywhere within the line. The search begins with the line following the current line and continues until a match is found or the end of file is reached. The located line becomes the current line.
LIST	List the entire file on the terminal or printer.

MERG N	Merge N lines from memory into the current file. The location (starting address) of the new lines will be requested and must be input from the keyboard. The new lines will be inserted following the current line. This command is designed for use with two files in memory, but it may also be used to move lines within a file if the destination is at a higher memory address than the source. If this is not the case, only one line at a time may be moved correctly within the file. The number of lines to be moved (N) must be input with the command, and is limited to a maximum value of 255.
N #	Move the current line pointer to the next line in the file (or # lines) and display the new current line. The optional number may be positive or negative.
P #	Print the current line (or # lines). The last line printed becomes the current line.
PAGE	Print one page (15 lines), beginning with the current line. The current line is unchanged by this command (it remains the first line printed).
R String	Replace the current line with the input string and display the result.
T	Set the line pointer to the top of the file and display the first line of the file.
TAPE	Output the current file to tape. The computer will respond with the question EXPAND FILE? which must be answered with a Y (for yes) or N (for no). If an expanded file is requested, all tabs contained in the file will be converted to the appropiate number of spaces before being written on tape. Data is transferred to the tape after the C/R following the Y or N answer is input, allowing time to set up the tape deck. The command may be aborted by inputting a question mark just before the carriage return. A 5 second leader is output before the data transfer is started.

Utility Commands - The utility commands allow displaying of the various pointers used by the program and specifying parameters to the program. All address output by these commands are presented in split octal, high order byte first. These commands interface with the terminal only, and will not output any data to the printer. The symbol * in the following command descriptions defines a letter identifier which must be input with the command.

DISP Display the current line pointer. This command is useful in conjunction with the MERG command to determine the address of the lines to be inserted.

DEOF Display the current end of file location.

DISM Display the current setting of the maximum memory size.

FORM Enter the formatter executive.

KILL * Set the kill character to *. This character, which is initialized to @, may be any printing character. The kill character deletes the entire input line if it is issued before the C/R is input.

Q Quit. Return to the system monitor.

RUBO * Set the rubout character to *. The rubout character, which is initialized to ", may be any printing character. The rubout erases the previous input character. Multiple rubouts may be used to erase several characters in the input line.

SETM Set the value of the maximum useable memory. This command does not verify that memory exists at the input address, so use care in specifying the maximum limit.

SETO * Set the output device to *. SETO P sets the output device to the printer, and SETO T sets the output device to the terminal.

TABS N Set the tab column spacing to N. N is limited to the range of 4 to 12.

TERM * Set the terminal to upper case only mode (TERM U), or upper and lower case (TERM L).

Error Messages - The program will output the error message WHAT? in response to unrecognizable or improperly formatted commands. In addition to this general error message, several other error messages may be displayed.

On all commands which require an address input, the address is tested against the minimum useable file address. If the input address is too low, the error message MIN ADDRESS (HL) XXX YYY will be displayed. This prevents overwriting of the program with the file being edited.

When a command is input which increases the size of the file, the new end of file location is tested against the current setting of the maximum allowable memory. If the maximum would be exceeded by execution of the command, the message OUT OF MEMORY will be displayed and the command will be terminated. The LOAD command does not test the maximum until after the load is complete. Data stored above the maximum limit can be overwritten by the LOAD command.

During execution of the LOAD command, the data which is stored is verified against the data which was read from tape. If a mismatch occurs, the message COMPARE ERROR AT (HL) XXX YYY will be displayed. This usually indicates a bad memory location at the displayed address. The LOAD command terminates if this message is displayed.

The MERG command checks the data to be transfered to determine if it is valid printing ASCII characters or control characters used by the program. If non-ASCII data is encountered, the message BAD DATA AT (HL) XXX YYY will be displayed. The MERG command terminates if this error message is displayed.

If execution of a command causes the end of file to become the current line, the message BOTTOM will be displayed, and the command will terminate. When this message is displayed as the result of a Find or Locate command, it indicates that the input string is not present in the portion of the file searched.

Special Features - The editor contains several special features designed to make usage of the program easier.

Tab characters may be input to generate equally spaced columns. This feature is useful for the generation of source code files, to define the columns for labels, op-codes, operands, and comments. The tabs are defined for equally spaced columns, as set by the TABS command. The tab column count is initialized to 6, which gives tab settings of 6, 12, 18, 24, etc.

If the load command is input as LOAD C, the input data from tape will only be verified against the data currently in memory. This command can be used to verify a tape that you have just written. This form of the command does not request a starting address, but rather begins the verification at the start of the current file. The tape should be running in the leader area when this command is issued, as it begins the verification immediately. The compare error message described for the load command will be output if the taped data does not match the data in memory.

Output to the terminal or printer may be interrupted by inputting a Control C. The Control C check is performed at the end of each line, and will always terminate the output at the end of a line.

Lower case input and output can be accomplished on an upper case only terminal through use of the TERM command for output, and the Control S key for input. For output, all upper to lower case shifts will be displayed as a Control L (displayed as ↑L), and all lower to upper case shifts will be displayed as Control U (↑U). Lower case characters will be transcribed and displayed as upper case. For input, each input of the Control S key will cause the editor to reverse the current status of the shift mode. If the current mode is upper case, the input will shift to lower case, and if the current mode is lower case, the shift will be back to upper case. Each shift will be displayed as ↑L for upper to lower case shifts, and as ↑U for lower to upper case shifts. The input mode will remained unchanged until another Control S is input or the editor re-enters the command mode (indicated by the > prompt). This shift mode can be rather confusing, but if you have an upper case only terminal, (as I do), and an upper and lower case printer, (also as I do), it provides a means of generating lower case files to be output to the printer.

As presently written, the program is set up to initialize to the upper case only output mode, and will accept either upper or lower case keyboard inputs. The SETO P command sets the output mode to an upper/lower case printer, and SETO T returns the mode to an upper case only terminal. You can ignore this feature if you have a lower case terminal and printer by using the command TERM L. This sets both the terminal and printer to output lower case characters unmodified. If the SETO T command is used, the TERM L command must be reentered to return the terminal to the lower case mode.

THE FORMATTER

The formatter processes a file generated by the editor and prints it in a neat manner. The formatter is entered by inputting the command FORM in the editor, and will respond with the message TEXT FORMATTER on the terminal. The formatter follows the general procedures described in the book "Software Tools", by Brian W. Kernighan and P. J. Plauger. This book is highly recommended to anyone who desires to study the details of implementing a text formatting program.

The formatter responds to commands imbedded in the text material to generate an output format of almost any style you may desire. In its simplest form, with no explicit formatting commands included in the text, the format will be defined by the initial values of the formatting parameters and the manner in which the material was entered into the file. Short lines will be combined to form longer lines, and long lines will be broken up to prevent them from overflowing the right margin. Extra spaces will be inserted between words to justify (line up) the right margin. Blank lines, or lines with leading spaces or tabs, will force a break in the output and cause a new paragraph to be started.

All of the formatting parameters may be modified by formatter commands. There are two levels of commands recognized by the formatter; executive commands and formatting commands.

Executive Commands - The executive commands are input from the terminal and specify the mode of printout to be used. The formatter outputs a less than symbol, (<), as a prompt, indicating that it is waiting for an executive command. The formatter recognizes the following seven executive commands:

PRNT Initialize the formatter, format the current file, and print it on the current output device. Unrecognizable commands are ignored. The printout continues until the end of file is reached, an end of record command is encountered in the text, or a Control C is input.

CHEK This command produces the same result as the PRNT command, except that unrecognized commands will be printed rather than ignored.

MORE Initialize the file pointers only and format and print the current file. None of the previously defined formatting parameters are modified by this command. This command is designed for printing multiple record files, where the formatting parameters are specified only in the first record.

CONT Continue formatting and printing of the file without performing any initialization. This command is intended for continuation of the printout after it has been interrupted by a Control C input.

PAUS n Set or clear the pause flag. If the pause flag is set, the formatter will stop after each page of output and display the messasge PAPER? on the terminal. Printing will resume when a carriage return is input. If you are using single sheet paper, this allows you to put a new piece of paper in the printer after each page of output. Clearing the pause flag inhibits this feature for use with a continuous form printer. The pause flag is not explicitly initialized by the formatter, so the PAUS command should be entered prior to printing the first file. PAUS 0 clears the flag, and PAUS or PAUS n, where n is nonzero, sets the flag.

SETO * Set the output device to the terminal (SETO T) or the printer (SETO P). The upper/lower case mode for the output is set as defined for the editor. The formatter will not accept the TERM command, so if you want to override the default settings of the SETO command, it is necessary to issue the SETO and TERM commands in the editor before entering the formatter. An error in this command will return you to the editor executive. This will be indicated by the editor prompt, (>), being output, rather than the formatter prompt (<).

EDIT Return to the editor.

When the formatter is first entered, none of the formatting parameters are initialized. The PRNT or CHEK commands perform this initialization. One of these two commands should be used to begin the printout. Multiple record files can be printed using the MORE command after the first record has been printed. The MORE command initializes the file pointers only, and will not modify any of the formatting parameters set in the first record. The records can be edited before printing by returning to the editor after each record is loaded. The editor does not modify any of the formatting parameters (unless the TABS command is used) except the file pointer, and it is reset by the MORE command. If you set the output device to the terminal for

editing, and back to the printer for the formatter, partially filled pages will be preserved.

Formatting Commands - The formatting commands are contained in the text to be printed, and allow the format to be modified dynamically as the file is printed. Formatting commands are entered on a separate line in the text. All of the formatting commands contain a period in column 1, and may be followed by a number if required. (Note: Sample formatting commands contained in this documentation for illustration purposes are indented from the left margin. In actual usage they would begin in column 1 and would not be printed in the formatted output.) A typical formatting command would appear in the file as

```
.RM 60
```

which would set the right margin to column 60. A leading plus or minus sign before the number in a command means that the affected parameter should be modified relative to its current value rather than setting it to the absolute value input. Thus, following the .RM 60 command with the command

```
.RM +10
```

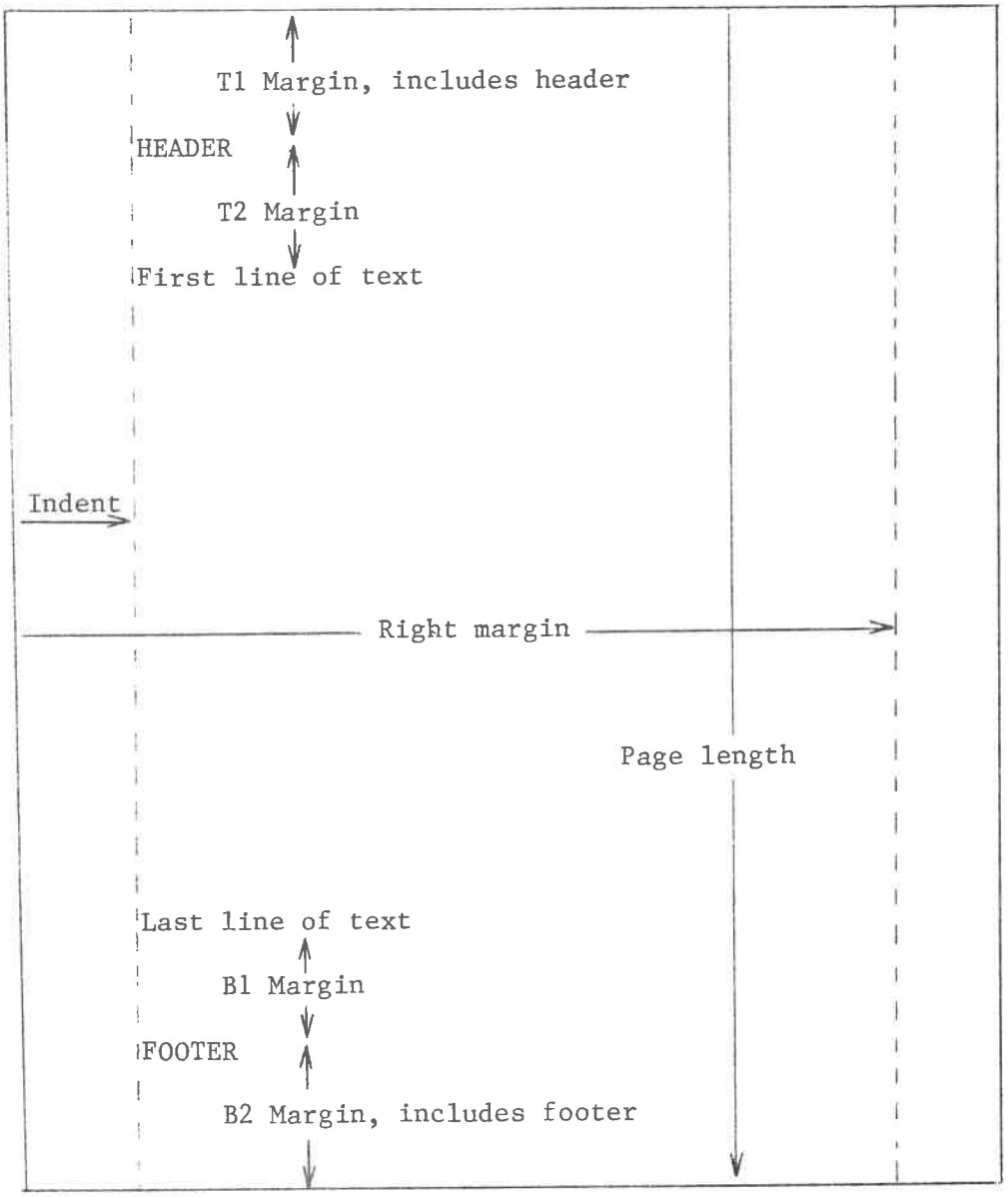
would set the right margin to column 70. If the command

```
.RM -10
```

were entered later, the margin would be set back to column 60.

The following figure illustrates the page size parameters which may be set by the formatter, and the table lists the commands which are available to control the output format. Those commands listed as causing a break will cause any partially filled lines to be output before the command is executed. Any optional parameters must be separated from the command by one or more spaces.

Page Size Parameters



Formatting Command Summary

Command	Break?	Default Value	Initial Value	Limit	Description
.B1 n	No	+1	2	*	Set bottom margin 1
.B2 n	No	+1	3	*	Set bottom margin 2
.BP n	Yes	+1	1	32767	Begin page n
.BR	Yes	-	-	-	Break
.CE n	Yes	1	0	255	Center n lines
.FI	Yes	-	On	-	Fill lines
.FO	No	-	Null	-	Footer title
.HE	No	-	Null	-	Header title
.IN n	Yes	+1	0	RM-1	Indent n spaces
.JU	Yes	-	On	-	Justify lines
.LS n	No	+1	1	15	Set line spacing
.NE n	Maybe	1	0	127	Need n lines
.NF	Yes	-	Off	-	No fill
.NJ	Yes	-	Off	-	No justify
.PL n	No	+1	58	254	Set page length
.RE	No	-	Off	-	Record end
.RM n	No	+1	72	120	Set right margin
.SP n	Yes	1	0	255	Space n lines
.T1 n	No	+1	1	*	Set Top margin 1
.T2 n	No	+1	2	*	Set Top margin 2
.TA N	No	-	6	12	Set tab column
.TI n	Yes	+5	0	RM	Temporary indent
.UL n	No	1	0	255	Underline n lines
.UW n	No	1	0	255	Underline words

The parameters are set to the listed initial values by the PRNT and CHEK commands. These numbers are a convenient set for using single sheet paper on a typewriter with 6 lines per inch vertical spacing and 12 characters per inch horizontal spacing. The header will be printed on the first line, with two blank lines separating it from the text. Two blank lines will separate the footer from the last line of text. There are 50 text lines on the page. Each line is 6 inches wide, and will have a justified right margin. The left margin is at the margin setting of the typewriter.

The default value listed is the value which will be used if no number is included with the command. The limit is the maximum value to which a parameter may be set. If a larger number is input with the command, or a relative offset causes the limit to be exceeded, the parameter will be set to the listed maximum value. No parameter may be set to a negative value.

The four commands .T1, .T2, .B1, and .B2 set the top and bottom margins of the page. The maximum limit for these commands is set to the current number of useable lines on the page. Thus, these commands interact with each other, and with the .PL (page length) command, to prevent a page size which has no useable text lines. Changing the line spacing with the .LS command does not affect the page size parameters. Multiple line spacings are counted as they occur to determine if the bottom of the page has been reached.

The .HE and .FO commands specify the titles to be placed at the top and bottom of the pages. Leading spaces in the titles are thrown away, and the titles begin at the left margin. To include leading spaces, such as to center the title, put a quote mark (") before the first desired space. If the number symbol (#) is included anywhere in the titles, this symbol will be replaced with the current page number when the title is printed. For example, the command

```
.FO "     Page #
```

will cause the bottom of each page to be labeled Page 1, Page 2, etc. The title will be indented 5 spaces from the left edge of the page, due to the 5 spaces included after the quote mark. If the T1 and B2 margins are set to zero, the titles will not be printed. Setting B2 to zero will override the pause flag setting, and eliminate the PAPER? request at the end of a page. Setting all four page length margins to zero will eliminate all page length formatting, resulting in continuous output with no titles.

The .IN command will move the left margin in on all following lines. A paragraph indent is generated using the .TI (temporary indent) command. This command will indent only the next line. The .TI n command produces a temporary indent n spaces in from the left side of the paper and the command .TI +n indents n spaces in from the current left margin. The two command sequences .IN 5 followed by .TI +5 and .IN 5 followed by .TI 10 produce exactly the same result. With the relative offset it is not necessary to remember the current setting of the left margin to produce a paragraph indent. The default setting produces a 5 space indent from the left margin if no number is input with the .TI command.

A hanging indent, such as to generate a numbered paragraph, can be generated by the command sequence

```
.IN 5
.TI -5
1.  This is an example of a hanging indent
produced by the formatter.
```

which will produce the following result:

```
1.  This is an example of a hanging indent
    produced by the formatter.
```

The .NJ command turns off the justify mode, resulting in lines which are only filled, with a ragged right margin. The .NF command turns off both the justify and the fill modes. Lines will then be output exactly as they were entered in the file. Only the page length formatting is in effect in the no fill mode.

The .SP n command will cause a break, and output n blank lines. To generate a break (output the current line, even if it is not completely filled), without generating a blank line, use .SP 0 or the break command, .BR. No right margin justification is performed on partially filled lines.

If the output is at the top of the page when the .SP command is encountered, the header and top margins will be output first. If the number input with the .SP n command causes the bottom of the page to be reached, the extra blank lines will be thrown away. The command

```
.SP 255
```

will always cause the bottom of the page to be reached, and the footer and bottom margins to be output. The same effect can be generated by using the .BP (begin page) command. The page number can also be modified using the .BP command, to begin with other than page 1, or to leave space for figures or photographs.

The .CE n command causes the next n lines to be centered. Each line will be output separately, and no justification will be performed on the centered lines. The line is centered between the current left margin (column 1 unless modified by a .IN command) and the right margin.

The .UL n command will cause the next n lines to be underlined. The .UW command (underline words) is similar except that the underlining is performed only on alpha-numeric characters. Spaces, punctuation, and symbols appearing in the

line will not be underlined when using the .UW command. The underline commands do not cause a break, so they can be used to underline words in the middle of a sentence by inputting the words to be underlined on a separate line in the file. Using the underline and center commands together will produce centered and underlined output. The underlining commands require a terminal or printer which has backspace capability, and should not be used if you do not have this feature on your system. The program outputs a control H, octal value 010, for the backspace.

The need command, .NE n, is used when you want to group n lines together on one page. If enough lines remain on the current page, nothing will be done by this command. Otherwise, the output will space to the bottom of the current page, allowing the group of lines to begin on the next page.

The .RE command signals the end of record of a multiple record file. When this command is encountered, the program will display the message END OF RECORD, START TAPE on the terminal and enter the tape load routine. The program will return to the formatter executive after the load is complete. As with the editor, the load will not begin until a carriage return is input, and inputting a question mark just before the carriage return will abort the command. This command does not cause a break, so the end of record can occur in the middle of a paragraph. If this command is omitted, and the end of file is encountered, the formater will clear the current page. Any partially filled lines will be output, as well as the bottom margins and the footer.

Restrictions and Special Cases - There are a few restrictions and special cases which can produce unexpected, and possibly undesired, results when using the formatter.

When printing the file using the CHEK command, improperly formatted or misspelled commands will cause a break and the bad commands will be printed. However, the extra line of output generated by the error printout will not be counted in determining if the page is full. This will cause the page to overflow the set length, and can produce some strange pages, especially with a form feed printer. This mode of operation was implemented intentionally, since if errors are to be printed they should be printed in an obvious manner. Usually I run the CHEK command with the output set to the CRT. This gives a rapid check, since the CRT display is much faster than the printer. With the output on the CRT, the page size parameters are meaningless anyway, so the overflow does no harm.

The CHEK command is also limited in the types of errors it will detect. It will catch misspelled commands, and commands

which are not followed by a space or carriage return, but it will not detect errors in the numerical data associated with a command. Thus the command error

```
.SP I5
```

instead of the desired command of .SP 15 will not be detected. Rather than outputting the desired 15 blank lines, this error would only cause a break, and no blank lines would be generated. The results generated by this type of error will vary depending on the command. In general it will only cause a break, (if the command would normally cause a break), and will not perform any other action.

When using the .NE command to group lines together, the current setting of the line spacing must be taken into account. If you want to group 10 lines together, and are using double spacing, the proper command is

```
.NE 19
```

This provides 10 lines for the text, and 9 blank lines to separate each text line. It is not necessary to reserve a line for the blank line following the last text line, unless you want to insure an increased spacing following the last output line.

When using hanging indents, using the commands .IN n followed by .TI -n, and using justified output, an undesired result can be produced by the extra spaces inserted to justify the right margin. This is best illustrated by the following command sequence,

```
.IN 5
.TI -5
1.  This illustrates an error which ocassionally
will happen when using hanging indents.
```

which produces the following result.

```
1.    This   illustrates   an   error   which
      occasionally   will   happen   when   using
      hanging indents.
```

The extra spaces inserted to justify the right margin have separated the number from the first word by more than the desired three spaces. The .TI -5 command forces the extra spaces to be inserted from the right end of the line, but if there are insufficient words in the line to accomodate all of the required spaces, they will overflow past the desired start of the line and generate the result shown. Rewording of the

first line can usually be used to prevent this from happening.

The formatter does not modify the spacing between words in the input line other than for right margin justification. Words can intentionally be spread apart by inputting extra spaces between them. One space is inserted after the last word in the input line, unless it is the end of a sentence, which will cause two spaces to be inserted. The end of a sentence is defined by a period, question mark, exclamation point, or colon (. ? ! :). If you want one of these special characters to be followed by only one space, the special character must not be the last character in the input line. Spaces after the last printing character in an input line are ignored, and will not modify the end of line spacing. A word longer than the line length will extend past the right margin, and long lines output in the no fill mode can also extend past the desired right margin.

Leading spaces or tabs in the input line produce the same result as the .TI command. A break will be generated, causing a new paragraph to be started. The next output line will be indented from the current left margin by the number of leading spaces in the input line. If a .TI command is followed by a line with leading spaces, the indent length will be controlled by the number of leading spaces rather than by the number associated with the command.

Tabs should normally be avoided in formatted text, except to generate paragraph indents or to print tables using the no fill mode. Tabs will be converted to the appropiate number of spaces based on their position in the unformatted input line, rather than where they would occur in the formatted output. Tabs can also cause overflow of the line buffer if they are included in a long input line. The formatter converts the tabs to spaces and stores the result in a line buffer before doing the formatting, and does not test for buffer overflow. Buffer overflow can cause strange and unpredictable results, and may require reloading of the program. Since it was felt that the probability of buffer overflow was low, the extra code required to handle it in an orderly manner was not included in the program. The line buffer is 132 characters wide, so buffer overflow problems can be avoided by keeping the input file line lengths below this limit.

RUNNING THE PROGRAM

The program will normally be supplied with a checksum routine for verification of the load operation. Assuming the program origin is as location 000 000 (split octal), the checksum routine begins at 024 367. Starting the program at this location will cause the checksum to be calculated and printed on the terminal. Execution will then be transferred to the editor, and any of the edit commands may be input.

The checksum value will vary, depending on the system configuration the program is set up for, and will be supplied with the program tape. The checksum routine is stored in the useable file area, and may be overwritten by the file. The checksum routine must be used before entering the main program, since the initialization sequence changes the checksum value. To bypass the checksum routine, start the program execution at location 000 000.

Taping of the file as soon as it is generated is recommended. A system crash or power failure which wipes out a large file can be very frustrating. If you have the file on tape, you can at least recover all but the latest changes.

When setting up a file to be processed by the formatter, it is best to input the formatting commands as the file is generated, rather than adding them after the main text has been entered. This may seem cumbersum at first, but it will usually produce better results. As you become familiar with the formatting commands, their use will become almost automatic.

This program is upward compatible with earlier version of the editor which I have written, and will process files generated by the earlier versions. The program has been tested and checked out to verify proper operation, but the possibility always remains that some bugs have slipped through undetected. Any errors found should be reported either to the distributor where the program was obtained, or directly to the author. If possible, a listing illustrating the error should be included. For formatting errors, this should include both the formatted output, and the original input file which was processed by the formatter. Speling errors are not to be considered program errors. The program is not that sophisticated at this time.

I/O Modifications

All I/O routines are external to the editor and must be supplied by the user. The I/O linkage is handled through the following jump table. The jump addresses given in the table are for my system, and will have to be changed for use on a different system. The jump table location assumes the program starting address is at 000 000, and will have to be adjusted accordingly if you have a relocated version of the program.

```
011 263   MONT   JMP  000 360   SYSTEM MONITOR
011 266   LODF   JMP  000 000   FILE LOAD
011 271   FILE   JMP  000 000   FILE OUTPUT
011 274   TAPO   JMP  036 361   BYTE OUTPUT TO TAPE
011 277   TAPI   JMP  274 360   BYTE INPUT FROM TAPE
011 302   PRNTR  JMP  240 362   CHARACTER OUTPUT TO PRINTER
011 305   TRMO   JMP  060 362   CHARACTER OUTPUT TO TERMINAL
011 310   TRMI   CALL 232 362   CHARACTER INPUT FROM KEYBOARD
                 ANI  177         STRIP PARITY
                 NOP              LEAVE ROOM FOR
                 NOP              ECHO IF
                 NOP              REQUIRED
                 RET
```

The I/O routines may modify the status flags (carry, zero, etc.), but should not modify any other registers. The input routines must return with the input data in the A register, and the output routines must output the data from the A register. The output routines must return the original output data unmodified. 38 bytes of stack space are available for use by the external I/O routines. Return to the editor should be accomplished by executing a RET instruction.

The program does not output a line feed following a carriage return. If your terminal or printer requires a separate line feed it will be necessary to test for the carriage return in the output routines and output the line feed before returning to the editor. Be sure to restore the carriage return in the A register before executing the return. Also, if your terminal requires the most significant bit of the output data to be set high, this must be done in the output routines. The program stores the ASCII data with this bit clear. The terminal input routine removes the most significant bit so it is not necessary to take any special steps for the input mode.

The keyboard input routine should automatically echo the input data back to the terminal prior to reentry to the editor. Alternately, the three NOP instructions shown in the input

routine can be replaced with a CALL TRMO instruction to accomplish the echo.

The tape output routine assumes that the output is in the "mark" state for generation of the 5 second leader at the start of taped output. If you are using a software UART routine, and the output port does not automatically initialize to the mark state, it is necessary to set the output to the mark state in the program. Four NOP instructions are included in the tape output routine (at location 000 214) for this purpose. A typical initialization routine to force the output to the mark state is

```
MVI    A,1
OUT    TAPE PORT
```

The only other change which may be required is in the Control C check routine. The following instruction is used for this purpose:

```
010 152    IN    1      READ KEYBOARD DATA
```

Change the input data port to correspond to the data port on your system if it is not port 1. Note that a status port is not used for this check, even for a serial input port.

The two file handling routines, LODF and FILE are presently unimplemented, and will restart the editor if they are used. These two routines are planned for file oriented tape or disk, and were included in the jump table for ease of modification of the program. The present tape routines input and output the data one byte at a time, with no leading or trailing sync bytes or checksums. The tape I/O stops when the end of file byte is detected. The general procedure for patching the program to file oriented tape I/O is as follows.

The command FILE is designed for output to file oriented tape. This command accesses the FILE address in the I/O jump table. The command is stored in the input buffer, and can contain up to 127 characters for parameters following the command. As with all of the editor commands, parameters must be separated from the command by one or more spaces. The variable IPNT (location 011 331) contains the address of the first non-space character following the command; the variable TOPL (location 011 333) contains the file start address; and the variable EFPN (location 011 337) contains the file end address.

The command LOAD F is designed for loading from file oriented tape, and will access the LODF entry in the I/O jump table. This command is also buffered, and may contain

additional parameters associated with the command. The variable IPNT again contains the address of the first non-space character following the command. Before returning to the editor, you should set the file starting address in the variables TOPL and PNTR (location 011 335), and store the end of file address in the variable EFPN.

For both the FILE and LOAD F commands, return to the editor should be accomplished by a RET instruction. All registers except the stack pointer may be modified by the external file handling routines, and need not be preserved for reentry to the program.

There are several approaches to allocating memory space for the I/O routines. On my system, the top 4K of memory space is dedicated to system monitor functions, and contains the I/O routines in read only memory. Scratch pad ram is also available in this space for patching in special purpose I/O routines. If you have an operating system in high memory, that is the ideal place for the I/O routines. The jump table can be patched to the correct locations for your routines with no further modifications required. For operating systems in low memory, relocation of the program is required, but no other modifications should be necessary.

The I/O routines can also be placed immediately following the program, eliminating the need for a separate operating system. Start the I/O routines at location 025 215 if you want to preserve the checksum routine, or at 024 367 to replace the checksum routine with the I/O routines. The minimum useable memory address should be changed to the new end of the program if this approach is used. This address is referenced only at location 006 250. Changing the data at that address to correspond to the end of the I/O routines will prevent overwriting of those routines by the file being edited.