SMOKE SIGNAL BROADCASTING

TEXT PROCESSOR

SSB TEXT PROCESSING SYSTEM

Smoke Signal Broadcasting
31336 Via Colinas
Westlake Village, CA 91362

COPYRIGHT NOTICE

WARRANTEE INFORMATION

The Text Processor is provided AS IS without warrantee. Reasonable care has been taken to insure that the Text Processor Software operates as described in the Text processor manual. If you do find a situation in which it does not operate as described, please contact us. We will attempt to correct any errors brought to our attention but we make no guarantee to do so.

IMPORTANT NOTE

This manual is supplied for use with both the 6800 version and the 6809 version of the Text Processor. On the disk supplied with the 6809 version, there is only one file. This file, named PRC.$, is for use with all versions of DOS69. On the disk supplied with 6800 versions of the Text Processor, there are 6 files. TXT35.$, TXT35A.$ and TXT35C.$ are for use with DOS68 versions 5 and 6 located in memory at $6080 through $7FFF, $A080 through $BFFF and $C080 through $DFFF respectively. PR.$, PRA.$ and PRC.$ are for use with DOS68 version 4 and earlier. (The commands .CC, .ES and .SW are not supported on this older version of the Text Processor).

CONTENTS

PREFACE

The SSB Text Processing System is one of the most complex programs released by SSB to date. With this in mind, the following recommendations should be noted by the user.

Do not expect to master the system with one reading of the manual. The entire document should be read lightly the first time through, followed by a more rigorous reading. The "Reference Manual" section is very concise and contains detailed descriptions of all of the commands of the processor. This is the section which should be studied extensively.

Since the system is so complex, many results may occur which are contrary to the user's intentions. If strange output is encountered, reread the sections of the manual covering the commands being used. As more experience is gained, the system will become an invaluable tool, but as with any complex system, it takes time to learn its operation.

INTRODUCTION TO TEXT PROCESSING.

This world is producing millions of words of text each day. There are words in newspapers, magazines, books, catalogs, pamphlets, letters, documents, and manuals, and they all need to be organized before publication. It would certainly be a never ending task if all text formatting and organization were to be done manually. It simply would not get done. Thanks to computers and programs called text processors, text formatting (sometimes called word processing) becomes a fairly trivial task. The text processor allows for convenient and precise page formatting and organization. The final copy becomes extremely readable and neat, which are desirable features of any printed matter.

Just what can be done with text processors? The simplest functions perform exact page fitting. In other words, if the text page should have one inch margins with a page number centered at the bottom of each page, and perhaps a special title at the top of each, the processor will automatically provide these, given the appropriate commands. Line justification is another feature provided. Several types are available which include left-hand justification (left edge straight, right edge ragged), right-hand only justification (left ragged, right straight), left and right (both edges are straight), and center justification (both edges ragged but lines centered). An extensive text processor will provide features which will allow special operations such as footnote processing. The SSB Text Processing System supports all of the above features.

To gain some insight into the use of a text processor, several specific examples will be given using the SSB Text Processor's command set. The commands used by text processors vary from system to system but many are used in the same fashion. The SSB Text Processor uses an intermixed command and text method. To issue a particular command to the processor, it is necessary to start the command in column number one of a new line and begin the command with the control character, a period ('.'). This is the method used by most of the large scale system formatters including NROFF*, which is the system the SSB Text Processor has been modeled after.

Before any specific examples are shown, a description of the 'environment' will be given. The environment refers to the basic page and formatting features which will be in effect unless otherwise specified. The initial or default environment is very important. The SSB processor, without any command information, will perform left and right justification with a line length of 65 characters (the standard 8 1/2" page line width). Page length is initially defined to be 66 lines which is the standard for 11" paper and 6 lines/inch spacing. Other initial environment features provide for the passing of blank lines to output, and

---

*NROFF is a text formatting program written at Bell Laboratories. It runs on many large operating systems, including the UNIX Time Sharing System.

for any line starting with a space or spaces to create a new line with the spaces now treated as unpaddable space characters*. With the environment initialized as above, it is possible to take any text file not having special command information embedded in it and still receive a sensibly formatted output. This is an important feature which is often overlooked by many processor designers. The environment may, of course, be changed or modified at any time by the use of special commands to allow for more personalized and detailed formatting jobs.

Let's take a look at some specific commands of the SSB processor. One of the simplest commands is the center lines command, .CE N, where N is the number of lines to be centered within the current line width. To use this command, as with any of the commands, it is only necessary to place the command right before the lines it is to affect. For example:

```
.CE 2
The Design of Text Processors
An Introduction
```

will cause the two lines listed to be neatly centered on the page. It can be seen that this is much easier than trying to manually calculate the correct spacing.

The initial environment was previously described. All of the parameters may be easily changed by the commands which directly affect them. One of the commands is .LN N and is used to set the current line length. To set the line length to 50, all that is necessary is a command line which reads as follows.

```
.LN 50
```

The length is now 50 columns. Another parameter easily set is the page length using the command .PL N, where N is the number of lines per page desired. Some other commands which change environment parameters include .FI and .NF which turn fill mode on and off (no fill) respectively. Fill means that as many words which will fit within the current line length are placed on each output line. This gives a straight left text edge and a slightly ragged right one. No fill simply copies the input lines directly to the output. It should be noted that 'fill' must be on for any justification to occur. The justification feature may be turned off using .NJ for 'no justification' or the type of justification may be set using .JU X. The X is the selection character and may be null which turns justification on in the mode it was previously defined, it may be R for right hand, C for centered, or N for normal (left + right). Left justification is obtained by turning 'fill' on and justification off.

---

*Unpaddable spaces are characters which appear as spaces on the output but are not recognized as such by the processor. This means these spaces will not be 'spread out' by the justification routines.

Another environmental parameter is the capitalization mode. This feature of the SSB Text Processing System allows an upper case only terminal to be used for preparation of text which will later be output on a hardcopy device having lower case capabilities. The commands .CP and .NC turn this feature on and off respectively. If this mode is on, all letters are automatically converted to lower case unless preceeded by a '@'. The '@' should be thought of as a typewriter shift key in its function. Another feature also enabled in this mode is similar to the 'shift and lock' on a typewriter. By typing a '^' all characters following will be upper case until another '^' is encountered.

It is often desirable, for readability, to use multiple spacing between lines. The SSB processor will allow this using the command .MS N where N is the space count desired and defaults to double spacing (N=2) if no value for N is given. The single space mode can be restored by either using .MS 1 or .SS for 'single space'.

Another group of commands deal with left margins and indentation. The left margin is normally set to Ø since the output device usually provides its own left margin (determined by paper positioning). Some applications require a wider margin at which time .LM N may be used to redefine it to be N spaces wide. Indent is similar to the left margin control with one difference. .LM N preserves the line length and simply moves the line to the right N spaces. .IN N, on the other hand, effectively reduces the line length by N columns in order to preserve the right hand margin. Setting the indent back to Ø will restore the full line length. Another form of indenting can be done by the use of the single indent command .SI N. Single indent is identical to indent except it is automatically restored to Ø after the line is output. It should be noted that the commands for left margin, indent, and single indent are additive in that if the following string of commands is issued:

```
.LM 1Ø
.IN 8
.SI 5
```

the resultant output line would be preceded by 23 spaces. Succeeding lines are preceded by 18 spaces assuming another .SI command was not used.

A note of caution is necessary concerning a characteristic of several of the processor's commands. Most commands will perform only their specified function but some also cause a line 'break'. A break is the forcing of output of the line currently being collected in the line buffer. Normally a line is kept in the buffer until the specified line length has been reached, at which time justification may or may not occur, depending on the mode enabled (also assuming that 'fill' is turned on). The break will cause the partial line to be output without being filled,

but the appropriate justification will be performed. This is useful for starting new paragraphs or new blocks of text. Some of the commands which cause a break are .CE, .FI, .NF, .IN, and .SI. Sometimes it is desirable that these commands do not cause a break. This can be done by using the 'no break' control character, ':'. So far, all commands have been shown preceeded by the normal control character, a period. To set an indent of 10 and not cause a break, the following should be used:

```
:IN 10
```

The colon may be used with any command, whether the command normally causes a break or not.

It is often necessary to produce a section of one or more blank lines. The space command, .SP N, can be used to output N blank lines. The space command also causes a break. If N is not specified, the processor will output 1 blank line. It may be required that the blank lines all be on the same page, maybe for later insertion of a photograph or illustration. The SSB Text Processor allows this by using the 'save space' command, .SV N, where N is the number of lines required. If there are not N lines remaining on the current page, no line is output but instead, printing continues and the count (N) is saved for later use. When the next page is reached, the 'output saved space' command may be used, .OS, which will then produce the remembered number of blank lines. A convenient method for using .OS will be given later. Another similar command is the 'need lines' command, .NL N, where N is a line count. This command says that there must be N lines remaining on the current page, and if there are not, eject to the next page. This is convenient for keeping special blocks of text together (keep them from being split by page boundaries), or for not starting a new paragraph at the bottom of a page if only 1 or 2 lines will fit.

The commands which have been described so far will allow very nice page formatting. If these were all that were available in a text processor, much time and effort would be saved. The SSB Text Processing System, however, offers many more commands and much more versatility. One feature often needed in documents or manuals is the page title. There are many different ways of providing titles but the SSB processor uses a title command which has the form:

```
.TL 'field1'field2'field3'
```

where field1 is left-adjusted, field2 is centered, and field3 is right-adjusted. Any one or all of the fields may be present. Another feature supported in the SSB processor is the ability to print the current page number in the text. Any place a percent sign (%) appears, it will be replaced by the page number. A few examples will clarify the use of the title command.

```
.TL 'Main Title'''
.TL ''Centered Title'Date'
.TL ''-%-''
```

The first line will left adjust "Main Title" on the page. The second example causes "Centered Title" to be centered and "Date" to right adjusted. The final example will cause the current page number to be printed between two dashes.

Now it would be nice if there was some way of getting the title (and maybe a few other commands) to automatically execute at the top and/or bottom of each page of output. The SSB processor offers two advanced features to perform this task: macros and traps. A macro is a set of commands grouped together and given a name. When this name is later referenced, the entire group of commands will be executed. Essentially, what results is the ability to write special programs using the command set of the processor to do specific tasks such as headers, paragraphs, special titles, etc. The trap allows the user to specify a certain line on the output page where a specific macro is to be executed. To solve the title problem stated above it is convenient to define two macros, a header macro and a footer macro. The purpose of the header is to perform a sequence of commands to make the top of each new page appear the same. The footer macro works at the bottom of each page. Suppose it was required that the top of each page have three blank lines followed by a centered title and the bottom of each is to contain a centered page number between dashes. The following macros and trap placement would satisfy this requirement.

```
.DM HD
:SP 3
.TL ''Page Title''
:SP 3
..

.DM FT
:SP 3
.TL ''-%-''
:PG
..

.AT 1 HD
.AT -7 FT
```

The .DM command is used to define a macro and the first one listed in the example defines the header macro called HD. The header macro will space down 3 lines (without causing a break since the no break control character (':') was used), print a centered title, and finally print 3 more blank lines without causing a break. The last line of the header macro definition is '..' and is the command for terminating a macro definition. The second macro defined is FT and is used for the footer. Upon execution it will space down 3 lines (without a break), print a

centered page number, and eject to the next page. The .AT commands were used to set the trap locations. .AT 1 HD causes the header macro to be executed at line 1 of every new page while .AT -7 FT causes the footer macro FT to be executed at the 7th line from the bottom of each page. The ability to specify trap locations and define macros makes titles and footers extremely simple and efficient.

One of the important aspects of using a text processor is the ability to make a few minor command changes and greatly change the final copy. As an example, suppose at the last minute it was decided that it would look better if there were four blank lines at the top of each page rather than three. If the document were being prepared by hand it would be necessary to retype the entire work to obtain the extra space. Using a small text processor it would only be necessary to go back and change the line count before each title. The SSB Text Processor and its ability to define macros means only one line in the entire text file needs to be changed. The second line of the header macro which is currently ':SP 3' would be changed to read ':SP 4'! One simple change and the desired result is obtained! It should be kept in mind that when preparing documents with a processor supporting macro capability, all of the often-used command strings should be defined in a macro so simple global changes may be easily performed if so desired.

There are more advanced features supported in the SSB Text Processing System. One of these is the ability to do conditional command execution. There are four forms of this command:

```
.IF O .XX
.IF E .XX
.IF N .XX
.IF !N .XX
```

where O and E stand for Odd and Even page number respectively, and N can be a number, a number register (to be explained shortly), or an expression containing numbers and number registers. The exclamation point is the 'NOT' operator and .XX is any command or macro name. The command works as follows; IF the condition is true (page is odd or even, or the number or expression is greater than zero) the command XX is executed, otherwise it is not. Preceding the expression by '!' will cause the command or macro to be executed only if the condition is not true (less than or equal to zero). The following special header macro definitions will illustrate the use of this command.

```
.DM HD
:SP 3
.IF O .TL '''Title'
.IF E .TL 'Title'''
:SP 2
..
```

```
.DM HD
:SP 3
.IF %-1 .TL ''Title''
:SP 2
..
```

The first header defined causes the title to be right-adjusted on odd numbered pages and left-adjusted on even pages. The second definition will print a centered title on each page except page number one since the value of the expression will be zero when the page number is one (remember that the '%' represents the current page number).

Another feature contained in the SSB processor is the ability to use number registers. Two types exist, one which allows the user to read and access certain system parameters including the date, page number, current indent, left margin, current column position, current line on the page, and line length. The second type are user definable and can be used exactly as variables would be used in a program. Number registers are the single letters A-Z and the percent sign (%) already introduced. Several other number register features are supported by the SSB processor, including auto increment, assigning values to the registers, use in expressions (as seen in the .IF command), and the ability to print any register on the output in either Arabic, capital Roman, or small Roman numerals.

Some processors, including SSB's, allow communication between the processor and the operator during actual text processing. Three of these commands take on the following form:

```
.ST
.TM any string
.GI any string
```

The first command will stop the processing and print 'STOP' on the user's terminal. This may be desirable if special paper positioning is required or other special action is needed. When the processor has been stopped it may be restarted by typing any character on the terminal except an 'S' which will halt processing. The second command listed will send 'any string' to the terminal as a special message. It may be used before the 'STOP' command to issue special instructions to the operator. The last command will 'Get Input' from the terminal and insert it into the output stream. 'Any string' can be used for a prompt. An example where this command is quite useful is in the preparation of form letters. The processor may prompt the operator for names and addresses which are then typed in at the terminal and automatically inserted into the text!

One final command will be described in this introduction, the 'divert text' command. Sometimes it is desirable to save text currently encountered for later use. An example of this is when trying to do footnotes. It would be nice if immediately

after the footnote reference was made, the actual footnote text could be typed, but saved for later insertion at the bottom of the page. The mechanism which allows this sort of operation is called a 'diversion' and is only available on the more complex text processors, such as SSB's. Two forms of the diversion usually exist:

```
.DI XX
.DA XX
```

where .DI instructs the processor to divert the following text into a diversion space named XX and .DA says to divert and append to the diversion space named XX. During diversion, all normal text processing still takes place, but rather than outputing the text to the printer, the text is written to a special place internal to the processor. The diversion process continues until the command for a divert is found without a name specifier. To recall the diverted text, it is only necessary to call it by name, exactly as macro calls are performed.

As an advanced exercise and demonstration of the diversion process (as well as many other processor commands) a complete set of macros for handling footnotes will be described. The reader should note that the following example is very complex and several readings will probably be required in order to fully understand its operation.

```
.NR B 7
.DM HD
:SP 2
.IF %-1 .TL 'FOOTNOTE TEST'''
:SP 2
.AU 1
.NR X Ø
.NR W Ø-#B
.IF #V .TR
.NS
..
.DM FO
.NR V Ø
.IF #X .FT
.CH FO -#B
:PG
..
.DM NM
.TL ''-%-''
..
.DM BF
.DA TX
.EV 1
.IF !#+X-1 .SA
..
```

- continued on next page -

```
.DM EF
.BR
.EV Ø
.DI
.NR W -#V
.CH FO #W
.IF #N-#P-#W .CH FO #N+1
..
.DM SA
--------------------
.BR
..
.DM TR
.BF
.NF
.FE
.FI
.EF
..
.DM FN
.DI FE
..
.DM FT
.EV 1
.NF
.TX
.RM TX
.DI
.FI
.EV Ø
..
.AT 1 HD
.AT -#B FO
.AT -4 NM
.CH FO 7Ø
.AT -#B FN
.CH FO -#B
.EV 1
.AU 1
.LN 55
.EV Ø
```

This example is quite similar to the one given in the "NROFF Users' Manual" written by J. Ossanna, of Bell Laboratories. To use these macros, merely insert their definitions at the beginning of the text file, and immediately after a footnote reference has been made, call macro BF. Following the call, simply type the footnote text and end it with a call to EF.

A description of the macros follows. The first line defines number register B and sets it equal to 7. Number register B is used to specify the size (in lines) of the bottom margin. A header macro definition follows (HD) and provides several functions. After spacing down two lines, the title is output

unless it is page number one (the IF command). Two more lines are produced and the auto increment value is set to one. Number register X is cleared and it is later used to keep track of the number of footnotes on the current page. Next, W is set to the location of the bottom margin trap and will later be adjusted as necessary if footnotes are added. The IF #V command checks to see if there is any remaining footnote text from the previous page and if so they are reprocessed (number register V contains the line count of the last diversion). Finally, the 'no space' mode is turned on to suppress any spaces which might otherwise get printed needlessly at the top of the page.

The footer macro, FO, clears the diversion count, V, and checks the value of X. If X is not zero (meaning there were footnotes on the page), macro FT is invoked. The footer is then restored to its original location by using the Change command as defined by B. The last command does a page eject. Macro NM is used to print a centered page number at the bottom of each page.

The begin footnote macro, BF, starts with a divert append into the diversion space called TX. The environment* is switched, and if it is the first footnote on the page, macro SA is invoked which outputs a set of dashes as a simple footnote separator line. Diversion of the footnote text continues until macro EF is called. At this time a 'break' is executed and the original environment is restored. The diversion is stopped with the DI command. Number register W is updated by the number of diverted lines and the footer trap line is changed to compensate for the added footnote lines. Finally, if the number of diverted lines was great enough to move the footer trap up past the current line position, the trap is reset to the next line. TR is responsible for rediverting any lines of footnote text which will not fit on the page. It is very unusual for this to happen but this may occur if a footnote is very long and is referenced near the bottom of the page.

Macro FT is used for reading back the diverted text. It switches environments, sets the no fill mode, and calls TX, the actual footnote text. TX is then removed from the macro list, the fill mode is restored, and the environment switched. The last group of lines is used to define the trap locations of the various macros. The header is set to line one, and NM is set to execute four lines from the bottom of the page. The trap for the footer is planted at -#B, then moved past the bottom of the page while FN is also placed at -#B. FO is then moved back as originally placed so in effect both FO and FN are placed at the same line, but trap FN can only occur if the footer trap is moved up by the occurrence of a footnote. The last lines switch to environment one and initialize it for a line length of 55 and auto increment of one.

---

*Environment switching is a feature supported by many of the larger text processors (including SSB's) which allows all of the major environment parameters to change simultaneously.

As a final example of how a text processor can be used, a sample section of text will be given. The text is shown first with the commands and then as the text processor would output the final copy.

```
.SP 2
.CE 2
^TEST OF SEVERAL^
^PROCESSOR COMMANDS^
.SP
.SI 5
@THIS EXAMPLE SHOWS HOW COMMANDS AND TEXT CAN BE INTERMIXED
FOR LATER PROCESSING BY A TEXT PROCESSOR.
@THE EXAMPLE STARTED BY CENTERING TWO LINES FOLLOWED
BY A SINGLE INDENT TO SIGNIFY THE START OF A PARAGRAPH.
@THE CAPITALIZATION MODE IS ON AND THE UPPER CASE SHIFT
CHARACTERS ARE BEING USED.
.SP
.LM 10
.LN 45
.JU C
@THE ADJUST MODE WAS JUST CHANGED TO CENTERING
AS WELL AS A LINE LENGTH OF 45.
@THE LEFT MARGIN WAS SET TO 10 TO GIVE A NICELY
CENTERED NARROW LINE.
@SPECIAL EFFECTS LIKE THESE ARE EASILY ACCOMPLISHED.
.SP
.LM 0
.LN 65
.JU N
@THE PARAMETERS WERE JUST SWITCHED BACK SO THE
LINE APPEARANCE WILL BE RESTORED.
THIS IS A SHORT EXAMPLE BUT SHOULD SHOW HOW THE
COMMANDS CAN BE INTEGRATED WITH THE TEXT.
```

This example appears in its expanded form on the next page.

This introduction to text processing is intended to be only that and is not a complete treatment of the subject. Many commands and features have been omitted. The ones included are the most general and the most used commands which offer the user a great deal of control and flexibility. Hopefully some eyes have been opened to the wide variety of applications of the text processor.

EXPANDED EXAMPLE

TEST OF SEVERAL
PROCESSOR COMMANDS

This example shows how commands and text can be intermixed for later processing by a text processor. The example started by centering two lines followed by a single indent to signify the start of a paragraph. The capitalization mode is on and the upper case shift characters are being used.

The adjust mode was just changed to centering as well as a line length of 45. The left margin was set to 1Ø to give a nicely centered narrow line. Special effects like these are easily accomplished.

The parameters were just switched back so the line appearance will be restored. This is a short example but should show how the commands can be integrated with the text.

*NOTE: This entire user's manual was prepared using the SSB Text Editing System and the SSB Text Processing System.

Command Form	Initial Value	Default Argument	Cause Break*	Explanation
I. PAGE CONTROL				
.PL +N	66 lines	66 lines	no	Page length.
.PG +N	N=1	-	yes	Eject to next page.
.PN +N	N=1	ignored	no	Page number.
.LM +N	N=0	previous	no	Left margin.
.NL N	-	N=1	no	Need N lines.
II. TEXT FILLING, ADJUSTING, AND CENTERING				
.BR	-	-	yes	Break buffer.
.FI	fill	-	yes	Fill output lines.
.NF	fill	-	yes	No fill or justification.
.JU C	jst,norm	just.	no	Justify on.
.NJ	just.	-	no	No justification.
.CE +N	off	N=1	yes	Center N input lines.
III. VERTICAL SPACING				
.MS N	prev	N=2	no	Multiple spacing.
.SS	single	-	no	Single space lines.
.SP N	-	N=1	yes	Space N lines.
.SV N	-	N=1	no	Save N lines.
.OS	-	-	no	Output saved lines.
.NS	space	-	no	No-space mode on.
.RS	-	-	no	Restore spacing.
IV. LINE LENGTH AND INDENTING				
.LN +N	65	prev	no	Line length.
.IN +N	N=0	prev	yes	Indent.
.SI +N	-	N=1	yes	Single indent.
.PI ST	-	-	yes	Put string in indent.
V. MACROS, DIVERSIONS, AND LINE TRAPS				
.DM XX	-	ignored	no	Define or redefine a macro.
.AM XX	-	ignored	no	Append to a macro.
.RM XX	-	ignored	no	Remove macro or diversion.
.DI XX	-	end	no	Divert out to macro "XX".
.DA XX	-	end	no	Divert and append to "XX".
.AT -N XX	-	-	no	Set trap at line N.
.CH -N -M	-	-	no	Change trap location.
.CH XX -M	-	-	no	" " "
..	-	-	no	End macro specification.

---

*The use of ':' as the control character (instead of '.') suppresses the break function.

Command Form	Initial Value	Default Argument	Cause Break	Explanation
VI. NUMBER REGISTERS				
.NR X +N	-	-	no	Number register.
.AU +N	0	prev	no	Set auto increment.
.AR	arabic	-	no	Arabic numbers.
.CR	arabic	-	no	Capital Roman numbers.
.SR	arabic	-	no	Small Roman numbers.
VII. TABS AND TAB CHARACTERS				
.TA N,..	none	none	no	Set tab columns.
.TF C	un.sp.*	un.sp.*	no	Set tab fill character.
.TC C	none	none	no	Set tab character.
VIII. THREE PART TITLES				
.TL 'left'center'right'			no	Define title.
.LT +N	65	prev	no	Length of title.
IX. CONDITIONAL INPUT COMMANDS				
.IF C COMMAND		-	no	If true, do command.
.IF !C COMMAND		-	no	"
.IF N COMMAND		-	no	"
.IF !N COMMAND		-	no	"
X. ENVIRONMENT SWITCHING				
.EV N	N=0	N=0	no	Change environments.
XI. SPECIAL CONTROL COMMANDS				
.CP	no caps	-	no	Capitals mode on.
.NC	no caps	-	no	No caps mode.
.ST	-	-	yes	Stop processing.
.EX	-	-	yes	Exit processor.
.PS	no pass	-	no	Pass text without proc.
.AP	-	-	yes	Repeat entire file.
.DH	-	-	yes	Double height line**.
.DW	-	-	yes	Double width line**.
.DB	-	-	yes	Double height and width**.
.SW	-	-	yes	Return to single width mode.

---

*Un.sp. = unpaddable space character.
**These commands require the output device
to support double dimensioned character printing.

Command Form	Initial Value	Default Argument	Cause Break	Explanation
XII. EXTERNAL COMMUNICATION				
.TM ST	-	-	no	Send string to terminal.
.GI ST	-	-	no	Get line from terminal.
XIII. MISCELLANEOUS				
.*	-	-	no	Comment field.
.CC N	-	-	no	Output control character.
.ES N	-	-	no	Output escape sequence.
XIV. UNDERLINE				
.UL	-	-	no	Underline next input line.
XV. DISK ORIENTED COMMANDS				
.IC C	">"	">"	no	Set item character.
.OF NAME	-	-	no	Open data file.
.CF	-	-	no	Close data file.
.RI S	-	-	no	Read item from file.
.NI N	-	N=1	no	Move to next item.
.NB N	-	N=1	no	Move to next block.

SPECIAL CHARACTER DEFINITIONS

Character	Meaning
\	Standard escape character.
@	Force capital letter.
^	Set capital letter mode.
#	Number register specifier.
.	Basic control character.
:	No break control character.

NUMBER REGISTERS

Register	Meaning
A-B	User definable
C	Current column count
D	Day of the month
E	End of data file flag
F	User definable
G	.GI & .RI character count
H	User def.
I	Current indent
J-K	User def.
L	Current line length
M	Month
N	Line count on page
O	Current left margin
P	Current page length
Q-U	User def.
V	Last diversion line count
W-X	User def.
Y	Year (2 digits)
Z	User def.
%	Page number.

REFERENCE MANUAL

INTRODUCTION

All input lines to the processor which are to be interpreted as commands should be started with the control character (a '.' or ':') in column one followed immediately by the two letter command. If the characters are not system command names or user defined macros, the line will be ignored. The 'nobreak' control character (':') may be used with any command to suppress normal line breakage during processing. Only a single command reference is permitted on any one line.

The following detailed command descriptions reference numerical arguments either as N, +N, or -N. N means any argument is taken as absolute and any previous value is simply replaced by the new value. +N is used when the argument may take any form of a number (either positive, negative, or absolute). Valid arguments of this form are +4, -1Ø, and 3 where the old value would be incremented by 4, decremented by 1Ø, and replaced by 3 respectively. Arguments of the form -N may use absolute values or negative values which are subtracted from the current page length (to reference N number of lines from the bottom of the page). When expressions are involved using the +N argument, the entire N is evaluated before the increment or decrement is applied (e.g. -6-3 will decrement the value by 3). Certain commands requiring arguments will keep the last argument assigned if the argument field is left empty when the command is called.

I. PAGE CONTROL

The page control commands are used to set the physical page parameters such as length, width, margins, numbering, etc. Top and bottom margins are not automatically provided and should be defined by the user with macros as described in a later section.

.PL +N — Set page length to N lines. Initial value is 66 lines and is reset to 66 if no argument is given. Does not cause a break. The maximum N is 255.

.PG +N — Eject to next page. If N is given the new page number will be adjusted accordingly. The page number is automatically incremanted if no argument is given and the command does cause a break. Max N is 255.

.PN +N — Set the page number to +N. If .PN occurs before the first break or first text, it will be set for the first page. The value is initially 1 and the command does not cause a break. The maximum page number is 255.

.LM +N — Set the left margin according to +N. The entire output line will be offset to the right by the number of spaces the current LM is defined. Initially there is

no margin (N=0) and no break occurs. Left margins should not exceed 100.

.NL N Need N lines on the page. If the distance to the next trap position or the bottom of the page is less than N, the paper is advanced to the next trap position (blank lines output). Otherwise no action takes place. No break occurs and the default argument is N=1.

II. TEXT FILLING, ADJUSTING, AND CENTERING

The following commands affect the appearance of individual lines of text. Two important parameters are referenced, Fill and Justify. The default fill mode is to fill output lines with as many words as possible without exceeding the set line length value. Any extra words are saved for output on the next line. A word is defined to be any string of characters separated by a space or spaces. If two words are to be separated by a space but are not to be split across line boundaries or separated by the justification routines, the unpaddable space character, "\ " (slash space) may be used. The default justification mode is left and right, giving straight margins on both sides. Filled lines which contain too few character positions to completely fill out the specified line length are padded with spaces until the correct length is achieved. The space filling or padding is done from alternate sides of the page as each line is justified to eliminate 'white rivers' which may otherwise occur in the text. No hyphenation is performed. It is important to note that fill must be on in order for the justification to be performed, but fill may be on by itself. If fill mode is off, characters are passed exactly as they appear on the input file.

.BR Break the line currently being filled in the buffer. The line is output after specified justification is done but no further filling or padding is attempted. Input lines beginning with spaces and empty text lines (blank lines) also cause a break.

.FI Fill mode is turned on and subsequent output lines are filled. This command causes a break.

.NF Turn off fill mode (nofill). Following input lines are neither filled or justified, but are copied to the output exactly as they appear on input, without regard to the current line length. Causes a line break.

.JU C Justification is enabled. If fill mode is off, adjusting will be deferred until it is back on. If the justify type character, "C", is present the justification type is set as follows: N sets for normal (default, left and right), R sets right only justify, and C will center lines (both margins ragged). If the type character is absent, justification is turned back on with the type previously used. No break is caused.

.NJ Turn justification off. If fill is on, the resultant output line will have a straight left and a ragged right edge. No break is caused and the justify type remains unchanged.

.CE +N Center the next N input lines. A break occurs before the command and then automatically after each line is output. If the resultant line is longer than the current line length, the output line will be left hand adjusted. The maximum count is 255.

III. VERTICAL SPACING

All line spacing defaults to standard single spacing. It may be set at any time by using the MS command. If the line spacing is N, N-1 blank lines are inserted after each output line. The occurrence of a trap will terminate any remaining spacing count. Contiguous space should be saved by using the SV and OS commands.

.MS N Set multiple line spacing to N. N-1 blank lines are inserted after each output line. No break is caused and if N is not specified the value of 2 will be used (double spacing). Max value is 255.

.SS Set single space mode. No blank lines are output after text lines and no break occurs.

.SP N Space N lines. The number of output lines is limited to the distance to the nearest trap or bottom of the page. If nospace mode is on, no spaces are output. If no value for N is given, it defaults to 1. SP causes a break.

.SV N Save N lines of space. If the distance to the next trap (or the bottom of the page) is greater than N, N lines are output, otherwise no lines are immediately output but the count (N) is saved for later output (see OS). Subsequent SV commands will overwrite any previously remembered N. Nospace mode has no effect. The command does not cause a break and the default value for N is 1.

.OS Output saved space. This command is used to output any previously saved space from the SV request. The remembered count is cleared after calling OS and nospace mode has no effect. A break does not occur.

.NS No-space mode is turned on. The no-space mode inhibits SP requests and PG requests without a next page number. This mode is automatically turned off after the output of a line of text. No break is caused.

.RS Restore space mode. If the nospace mode is on, it is turned off with this command without causing a break.

IV. LINE LENGTH AND INDENTING

Using the following set of commands, the user has complete control over the line length and various forms of indenting. The line length includes all indent spaces but does not include left margin spacing. As long as the fill mode is turned on, the resultant output line will be less than or equal to the current line length minus the indent. Line lengths of less than 6 columns are not permitted.

.LN +N — Set line length. The initial value is 65 columns and the command does not cause a line break. Line lengths must be between 6 and 255 columns inclusive.

.IN +N — Set the line indent according to N. With a line length of L and an indent of N, N spaces are output before each line and the remaining text is restricted to a size of L-N. Initially the indent is 0 and the command causes a break.

.SI +N — Single indent N spaces. Only the next output line will be indented by the amount specified by N. Note that single indenting may be done backwards into an indent field. (e.g. if indent is 10, SI -4 would temporarily set the overall indent to 10-4 or 6). IN and SI counts are cumulative and the final value may not be negative! This command causes a line break.

.PI ST — Put string in indent field. The string represented by "ST" (leading spaces ignored), is inserted into the field normally filled with spaces by the indent count. If the string is longer than the indent count, the string will be truncated so it will not extend past the indent field.

V. MACROS, DIVERSIONS, AND LINE TRAPS

A macro is a set of commands and/or text which can be assigned a name and called by name at a later time. All macro names are two characters long and must be different from any names already in existence in the system command name table. Macros are defined or redefined by using the DM command, or by using the output diversion command, DI. Macros already in existence may be appended to by using the AM or DA commands. If a macro is named XX, it may be invoked by an input line beginning with ".XX". A trap may also be placed at a specific vertical page placement to cause automatic macro execution at that point by using the AT command. During macro definition, number registers are not expanded into numeric values but are at the time the macro is executed. No other special character translation is done during macro definitions (e.g. tab expansion, etc.). Keep in mind that macros may be any combination of commands, macro calls, and text, but a macro may not define another macro (it may create a diversion).

A diversion is treated as a macro upon execution but is created in a different manner. Processed output may be diverted into a macro space for such purposes as footnote processing or vertical page size determination for conditional changing of page parameters (number register V contains the last diversion line count). All normal processing takes place during a diversion except left margins. It is standard practice to read back the diverted text in 'nofill' mode to suppress further line processing.

If at any time during macro definitions or diversion creation the macro space is overflowed, a system error will be generated and processing will be halted. None of the macro commands cause breaks in the line filling.

.DM XX Define or redefine a macro with the character name XX. The actual macro begins with the next input line. The macro definition is copied until the termination character ".." is found starting in column 1. Macros may not contain DM requests but may create diversions.

.AM XX Append to the macro named XX. This command acts exactly like DM except the following input lines are appended to an existing macro rather than creating a new named space.

.RM XX Remove macro or diversion. The macro named XX is removed from the name list and subsequent calls to this name will have no effect.

.DI XX Divert output into the macro space named XX. The macro named XX is defined or redefined at this point. All normal text processing occurs during diversions except left margin page offsetting is not done. The diversion process is ended when another DI or DA is encountered. Diversions can not be nested! The count of the number of lines last diverted is kept in number register V for possible later reference.

.DA XX Divert append version of DI. The same rules apply for both commands.

.AT -N XX At line N invoke macro XX. Any macro previously planted at line -N is replaced by XX. N is measured from the top of the page (Ø or 1 may be used to represent the top) and -N is measured from the bottom of the page (e.g. if the page length is 66, line -1 represents line 66). If no macro name is given with the command, the trap located at line -N, if any, is removed.

.CH -N -M Change trap. See next.

.CH XX -M Change the trap planted at line -N to occur instead at line -M. Alternately, change the location of the trap for macro XX to line -M. If there is not a trap set at -N, the request is ignored.

.. Terminate a macro definition.

VI. NUMBER REGISTERS

Number registers are a type of variable used during processing. There are two classifications, user definable and system. Number registers have single character names (A through Z and '%'). Number registers may be used any time a number is expected in a command and also may appear imbedded in text. There are two methods of referencing a number register:

```
#X
#+X
```

where '#' is the register designator character and X is the name of the register. When using '%' it should not be preceeded by the '#'. The '+' in the second example specifies that the number register is to be auto incremented prior to its use and it will retain the new incremented value. The auto increment amount is set using the AU command. When a number register reference is encountered it is converted to decimal, lower case Roman, or upper case Roman, as determined by the mode set. Number registers appearing in macro definitions are not converted until the macro is actually executed. Number registers may also be used to construct expressions any time a number is expected in a command (expressions may not be imbedded in text). The expressions are evaluated left to right and may contain only the operators '+' and '-'.

.NR X +N Assign a value to number register X. This command should only be used to assign values to user definable number registers.

.AU +N Set the auto increment amount to +N. Any time a register is referenced as "#+X", the AU value will be added to it prior to its actual use.

.AR Arabic numbers. See below.

.CR Capital (upper case) Roman numbers. See below.

.SR Small (lower case) Roman numbers. Number registers will subsequently be converted into Arabic, capital Roman, or small Roman respectively. This mode is initially Arabic and also applies to the outputting of page numbers using the '%'.

The following is a list of the system and user definable number register names.

Register	Meaning
A-B	User definable
C	Current column count
D	Day of the month
E-F	User def.
G	Get input (.GI) character count
H	User def.
I	Current indent
J-K	User def.
L	Current line length
M	Month
N	Line count on page
O	Current left margin
P	Current page length
Q-U	User def.
V	Last diversion line count
W-X	User def.
Y	Year (2 digits)
Z	User def.
%	Page number

VII. TABS AND TAB CHARACTERS

The currently defined horizontal tab character is replaced by the required number of fill characters corresponding to the distance to the next defined tab stop column (on the line currently being filled). The fill character is normally the unpaddable space character but may be defined by using the TF command. Up to 20 tab stops may be defined and should be set in ascending order. Initially no tab stops are defined and the tab character is null. Any non alphanumeric character may be defined as the tab character. It should be noted that using tabs with the fill mode turned on can result in nonsensical output tab fields since the user may not know what the current output column is.

.TA N,.. Tab stop settings. The default tab stops are all null (none) and a total of 20 may be defined. The stop values may be separated by spaces, commas, or any other nonnumerics, e.g. TA 10,20,25,40.

.TF C Set the tab fill character. This is normally the unpaddable space character but may be defined to any nonnumeric printable character. If 'C' is not specified the fill defaults to the unpaddable space character.

.TC C Define the tab character. Initially the tab character is null (none) but may be defined to any nonnumeric printable character. If 'C' is not specified the tab character again becomes null.

VIII. THREE PART TITLES

Very convenient titling may be performed by using the TL command. Three fields may be used for left, centered, and right justification of titles. All 3 fields may be used or any combination of fields. The justification is done with respect to the title length which is independent of the defined line length. This length is initially 65 columns. The use of TL has no effect on current line accumulation (does not cause a break). .TL is usually used in header and footer macros. For example, .TL ''-%-'' will print the page number in the center of the title length.

.TL 'LEFT'CENTER'RIGHT'
: Place titles adjusted according to field. The strings represented by "LEFT", "CENTER", and "RIGHT" are respectively left adjusted, centered, and right adjusted within the current title length. Any of the fields may be empty and any nonnumeric printing character may be used in place of the field delimiter "'". The "%" character will be replaced by the current page number in Arabic or Roman representation.

.LT +N
: Set title length. The lengths of titles and lines are separate parameters. Indents do not apply to titles but left margin adjustment does.

IX. CONDITIONAL INPUT COMMANDS

Input command and macro calls may be performed on a conditional bases. Chained conditionals are permitted as in: IF #A IF #B .XX.

.IF C COMMAND See next

.IF !C COMMAND "

.IF N COMMAND "

.IF !N COMMAND
: IF is the conditional command. "COMMAND" can be any system command or macro name. "C" is a built in condition code and can be either O or E to represent Odd or Even page numbers respectively. "N" is any number and can be a number, a number register, or any combination of these in the form of an expression using addition and subtraction. If the condition is true (the built in condition is satisfied or the number is greater than zero), the command or macro named is executed, otherwise the command is ignored. If "C" or "N" are preceeded by a '!' (not), the command is executed if the condition is false or the number is less than or equal to zero.

X. ENVIRONMENT SWITCHING

There are a number of parameters which control the text processing and are grouped together and called the environment. These environment parameters may be changed all at once using the switch command. There are two environments, Ø and 1. They both have identical initial values for all parameters. Parameters within these environments are those associated with:

line length	vertical line spacing
indenting	centering count
adjusting	auto increment
filling	partially collected words
title length	partially collected lines

All other parameters are global, or in other words, they are not switched with the environment but remain unchanged. Examples of global values include left margin, page number, current line number, number registers, trap tables, and macro definitions. Since partially collected words and lines are kept with the environment, switching environments will not cause a break and will also preserve any left over words.

.EV N — Change to environment N where N can be Ø or 1. If N is left null, environment Ø is assumed.

XI. SPECIAL CONTROL COMMANDS

The following commands control certain aspects of the processor. The double height and width commands are hardware dependent. You should refer to the "adaption" section of this manual for details.

.CP — Turn capital letter mode on. When enabled, this mode will allow the use of an upper case only terminal to prepare text for later output to a device which supports both upper and lower case. Each character is automatically converted to lower case unless it is immediately preceeded by a '@' at which time that character remains upper case. Strings of characters may be kept in upper case by enclosing them between up arrows "^". The "@" is like a shift key and the "^" acts like a shift and lock key.

.NC — Turn off capitals mode. Initially this mode is off and the special capitalization characters ("@" and "^") are ignored.

.ST — Stop causes processing to temporarily halt and the word "STOP" is output to the terminal. At this time, typing an "S" will cause all processing to be stopped and the processor will be exited. Typing any other character will cause processing to continue. The stop command does cause a line break.

.EX Exit the processor. Text processing is stopped just as if all input had been finished. This command is useful in conjunction with the IF command.

.PS Pass all input to the output. This command is primarily intended as a debugging aid since it allows all input (including command lines) to be passed to the output. No command interpretation or processing is done and once in this mode, the remaining text will be passed until the end of the input file is reached.

.RP Repeat processing on file. This command will cause the file to be 'rewound' and all processing to be repeated. This is useful for some form letter type applications.

.DH Print the next line in double height characters. This feature requires special hardware on the output device. Consult "Adaptions" for details.

.DW Print the next line in double width characters. Requires special hardware.

.DB Print next line in both double height and double width characters. Requires special hardware.

.SW Single Wide Command allows the user to turn OFF Double Wide printing. Some printers do this automatically upon receipt of a carriage return. Others require the use of this command.

XII. EXTERNAL COMMUNICATION

Two commands exist which allow for communication between the processor and the user during actual text processing. The TM command is useful for sending special instructions to the terminal such as paper adjustment or character font change information. The GI command can be used in form letter preparation or insertion of special text strings while processing is taking place.

.TM ST Send a message to the terminal. ST may be any string of characters or words. The leading blanks are ignored. The message is simply output to the terminal and may be used before the Stop command to issue special instructions.

.GI ST Get input from the terminal. If ST is present (any string), it is output to the terminal as a prompt message. Characters typed from the terminal following the execution of GI are automatically inserted into the input stream for text processing. This command can be used to get name and address information for form letter preparation. The 'get input' function is terminated by typing a carriage return, therefore, only

one line of text may be entered with each GI command executed. After completion of the command, the number register G contains the character count of the string typed (not including the carriage return).

XIII. MISCELLANEOUS

The following describe some of the smaller features of the text processor.

.* Comment field. This may be used to insert comments into the input text and will be ignored by the processor. No output is created with this command (the comment is not passed to the output).

.CC N Control Character Command allows the user to output any control character between hexadecimal $00 through $7f. N is represented as a decimal number between 0 and 127. The use of this command is primarily associated with intelligent printers that can be programmed by sending certain control character sequences to them.

.ES N Escape Sequence Command allows the user to output any escape sequence (the ESC character plus another following character) between hexadecimal $00 thru $7f. N is represented as a decimal number between 0 and 127. Again, the use of this command is primarily associated with intelligent printers that can be programmed by sending certain escape sequences to them.

Special Characters

\ Standard escape character. This character is used to remove special meaning from a character. For example, if a percent sign ("%") is needed in the output it is necessary to preceed it with the "\", otherwise it will be interpreted as the page number (e.g. \%). To print a backslash, "\\" must be used.

@ Force upper case letter if in the capitals mode (CP). This acts similar to the 'shift' key on a typewriter. Example: "@test" will be output with an upper case "T" and lower case "est".

^ Upper case string delimiter. This character acts similar to the 'shift and lock' key on a typewriter. As an example, ^this is a test^ would cause "this is a test" to be output in all upper case characters. The capitals mode must be on (CP).

Number register specifier. When an alphabetic character is immediately preceeded by a "#" it will be interpreted as a number register. Example: "#A" refers to number register "A".

. The period is the basic command control character. If in column one, it specifies a two character command or macro name follows.

: The colon is the no-break control character. It functions exactly like the period, but will suppress breaks caused by various commands.

% Page number symbol. Any place the percent sign appears, it will automatically be replaced by the current page number.

Special notes

A. Any time input is being typed into the processor, typing a 'control X' will delete that line and issue a "?" as a prompt.
B. The processor automatically makes sure there are two spaces after ".", "!", or "?". This does not apply to punctuation immediately followed by another character.

XIV. UNDERLINE

The following command permits underlining of words but may only be used with printer devices which support single character backspace capability. Unpredictable results will occur when trying to use this command on printers not supporting backspace.

.UL Underline the next input line. The following line of text (single or multiple words) will result in the output being underlined. Only alphanumeric characters are underlined.

XV. DISK ORIENTED COMMANDS

The following commands deal with the use of a "data file". The data file is a set of "blocks", with each block being divided into "items". An item can be any set of text or processor commands followed by an "end of item" character. The "end of item" character is initially a '>' but may be redefined using the .IC command (see below). The end of a block is specified by a null or empty item (two successive end of item characters form a null item; e.g. End of block>>) There are processor commands which allow inserting items into text (see .RI), skipping items (see .NI), moving to a new block (.NB), and the ability to open and close data files. For a specific example of these commands, see the Form Letter example in the MACRO LIBRARY section.

.IC C Set the end of item character. This character is initially a '>' but may be defined to any nonalphanumeric printable character. If 'C' is not specified, the character defaults back to a '>'.

.OF NAME Open a data file. This command will prepare the specified file for reading. If 'NAME' is specified on the command line (it should follow standard file spec rules) that file will be opened if found on the disk. If 'NAME' is not specified on the command line, the processor will prompt the terminal with: "DATA FILE NAME? " at which time the desired file name should be entered. If a file is already open, the .OF command will be ignored by the processor. It is only possible to have one file open at any one time. Closing a file using .CF will allow another file to then be opened.

.CF Close data file. If a data file is opened, it will be closed and not allow any more data to be read from it. If no file is open, the command has no affect.

.RI Read item from input file. If a file has been opened, the RI command will cause input from the file until an "end of item" character is read. The end of item character will be returned as a space if in the fill mode, or a carriage return if in nofill mode. If an S appears on the calling command line (.RI S), no

character will be returned for the end of item character. In other words, the character will be 'S'uppressed. If there are no items remaining in the current block, .RI will have no affect. The RI command will also be ignored if no file has been opened. After reading data with the RI command, number register G will contain a count of the number of characters just read in.

.NI N — Move to next item. Normally sequential items are read by using the RI command. It is often desirable to skip items while processing text from a data file. The .NI command is used to skip one or more items in a block. If N is present on the calling line, it should be a number (or number register) which specifies the number of items to be skipped. If N is not present, the default is one item to be skipped. NI will not move past the end of a block.

.NB N — Move to next block. The use of NI and RI commands cause the sequential reading of items and will never move into the next block. It is necessary to use the .NB command to advance to the next block. If N is specified (a number or number register), N-1 blocks will be skipped. (example: If .NB 2 were specified, the next block would be skipped over and the next data read would be from the block following). If N is not specified, it defaults to 1. If there are no more blocks left in the data file and the .NB command is used, number register E will be set to one to designate an End of file condition.

USING THE TEXT PROCESSOR

I. BRINGING UP THE SYSTEM

The disk processor command file name is "PR". The general syntax for the PR command is:

PR,<file spec>[,<list of file specs>]

The <file spec> designates which text file is to be processed. If the text to be processed is divided among several files, each file spec may be listed separately on the calling line separated by commas. A special feature supported by PR is the ability to process files from any number of discs on systems containing a limited number of drives. Substituting a '*' for the <file spec> any where on the calling line where a <file spec> is expected will cause the processor to halt and output to the terminal:

CHANGE DISKS AND TYPE A KEY

At this time, insert the disk containing the continuation file(s) and type any key to restart processing. It should be noted that the ability to process multiple files with one calling line should only be used when the files are actual continuations of the same text. The processor treats them as if they were all part of the same file, continuing page numbers, indenting, page width, etc., just as if the first file had never ended.

Another feature supported by the processor is the ability to automatically process a macro definition file prior to processing any of the files specified. Upon the execution of PR, the disk in drive Ø is searched for a file named 'MACRO'. If none is found, the processor starts processing the first file specified. If a MACRO file is present, it is read in and processed, just as if it had been the first file specified in the calling line. This is useful for defining all often used macros in this file so it is not necessary to redefine them in each processor text file prepared.

A few examples will clarify the calling of PR:

```
PR,CHPTR1
PR,Ø:CHPTR1,1:CHPTR2,*,Ø:CHPTR3
```

The first example will process the file named CHPTR1. The file MACRO will also be processed if it exists. The second example will first try to process the file MACRO, then process the files CHPTR1 on drive Ø and CHPTR2 on drive 1. The processor will then halt and output the 'CHANGE DISK' message to the terminal because of the '*' used as a file spec. After changing disks in drive Ø and typing a key, the processor will process the file named CHPTR3 on drive Ø.

When the procesor is called, the following message will be output to the terminal:

PAGE LIMITS?

and is used to specify a particular block of pages to be processed. Typing a carriage return will cause all pages to be processed and output. Typing two numbers separated by a space or a comma will cause only the pages between those numbers (inclusive) to be output. For example, typing:

10,16

will result in only pages numbered 10 through 16 to be output. If just one number is entered, the processor will start outputing at that page number and continue to the end of the file. It should be noted that the processor always starts numbering the first page as number one unless instructed otherwise. As the processor is working, it may be stopped at any time by typing a "control C" on the terminal. (This feature is only supported on computers using a serial type interface (MP-S) as the terminal interface port.) The processor will respond with:

..BREAK..

output to the terminal. At this time processing may be continued by typing any character except an "S" which will cause the processor to be exited.

II. GENERAL USE

There are several things to keep in mind while preparing text for the text processor. Remember that all commands must begin in column one. It is usually most convenient to begin each sentence on a new line for easy future editing. Macros should be used as often as possible. The reason for this is to keep global changes as simple as possible, e.g. change only one line in a macro as opposed to changing single commands scattered throughout the file. It is not necessary to understand how the macros provided in this manual work in order to use them. All that is necessary is to know how to use them which is thoroughly explained. As experience is gained with the processor, you will be able to create your own special purpose macros for easy formatting.

MACRO LIBRARY

The following macro descriptions range from simple header and footer macros to a very complex footnote macro. It is not necessary to understand how the macros work, just how to use them. Each macro includes a description of what it does and how it can be used.

I. HEADERS AND FOOTERS

These macros are used to define top and bottom margins and also specify the contents of these margins, such as page numbers, titles, etc. Almost all processing jobs will require some sort of header and footer. Usually the macro definitions are placed at the beginning of the file (they need to appear before they are called for execution). The "AT" command is used to set the trap location (the line at which the macro should automatically execute) of each of the macros. Headers are set to line 1 and footers to a specific distance from the bottom of the page. Once these macros have been defined and their trap locations set, they can be forgotten about since the processor will do all the rest of the work. The first macro is a simple header macro which provides two blank lines, a centered title, and two more blank lines at the top of each page.

```
.DM HD
:SP 2
.TL ''CENTERED TITLE''
:SP 2
.NS
.OS
..

.AT 1 HD
```

All of the header macros will contain a NS and OS command. NS will suppress any unnecessary spacing which may occur due to the unpredicted appearance of a SP command. For example, if the start of a new paragraph just happens to start at the top of a new page, there is no reason for the paragraph macro to space down two lines, since we are at the top of the page. NS will keep this from happening. The OS command instructs the processor to output any 'saved space' from the previous page. The next header is a little fancier. It does everything the previous one does except the titling is done a little differently. Here, if the current page number is even, the title is left hand justified. If the page is odd, the title is right hand adjusted.

```
.DM HD
:SP 2
.IF E .TL 'EVEN TITLE'''
.IF O .TL '''ODD TITLE'
:SP 2
.NS
.OS
..

.AT 1 HD
```

Subtitles may be used by simply placing a second TL command which contains the subtitle. The last header example is for those using a printer which uses separate sheets of paper (as opposed to continuous fed). This macro will issue a message to the terminal which instructs the operator to insert a new sheet of paper, before each page of text is processed. The paper should be set up such that the first line printed will be the top line of the paper. The operator will have to type a character on the terminal after each stop to restart the processor. Remember that typing an "S" will halt the processor.

```
.DM HD
.TM INSERT NEW SHEET
:ST
:SP 2
.TL ''TITLE''
:SP 2
.NS
.OS
..

.AT 1 HD
```

Footer macros are similar to headers except they are set to execute at the bottom of a page. For example, specifying AT -6 FO would cause the macro called FO to automatically execute at the 6th line from the bottom of the page. The first footer gives a five line bottom margin with the page number between 2 dashes centered on the page, 3 lines from the bottom.

```
.DM FO
:SP 2
.TL ''-%-''
:PG
..

.AT -5 FO
```

It is often desirable to have page numbers on every page except page number one. The following footer will do exactly that.

```
.DM FO
:SP 2
.IF %-1 .TL ''-%-''
:PG
..

.AT -5 FO
```

There are several other types of header and footer macros which can be created. Some of these appear in the macros which follow.

II. PARAGRAPHS AND HEADINGS

There are many forms of paragraphing. The SSB Text Processor does not restrict one to using one particular form. One type of paragraph is to produce one blank line and start the first line of the paragraph indented five spaces. The following macro does just that:

```
.DM PP
.SP
.SI 5
..
```

To use the paragraph macro, simply call it by name any time a new paragraph is desired (e.g. type ".PP" in column one). One little feature which may be added to the macro is a need lines command, NL. In the following example, NL 3 is used to tell the processor that we desire at least three lines be left on the page before a new paragraph is started. This will keep one or two lone lines from being placed at the bottom of the page.

```
.DM PP
.SP
.NL 3
.SI 5
..
```

Many other types of paragraph macros may be created along the same lines as those presented.

Another useful macro can be created for major heading creation. One type of major heading might have a centered title spaced two lines down from the last line of text. The macro to accomplish this may look as follows:

```
.DM MH
.SP 2
.CE
..
```

To use this macro, type ".MH" when the heading is desired. The next line should contain the heading title. For example:

```
Line of text.
.MH
Heading Title
```

The last two macro examples are quite simple, but show how even two or three lines of commands may be replaced by a single macro call. This is quite useful if these operations are going to be repeated many times throughout a document.

III. FOOTNOTES

The following set of macros is all that is required to do very efficient and easy footnote handling. A description of how they actually work is contained in the introduction of this manual. To use these macros, it is only necessary to include their descriptions at the beginning of your file. As soon after a footnote is referenced in the text, call the macro BF (begin footnote) to begin the footnote. Immediately following this call, type the contents of the footnote, followed by a call to the macro EF (end footnote). The following serves as an example:

```
Text here referencing a footnote*.
.BF
*Footnote contents typed here and
may be several lines long.
.EF
```

It should be noted that the footnote macros contain their own header and footer macros which may be modified as desired. These macros should be the first lines of a file.

```
.NR B 7
.DM HD
:SP 2
.IF %-1 .TL 'FOOTNOTE TEST'''
:SP 2
.AU 1
.NR X 0
.NR W 0-#B
.IF #V .TR
.NS
..
.DM FO
.NR V 0
.IF #X .FT
.CH FO -#B
:PG
..
.DM NM
.TL ''-%-''
..
```

- continued -

```
.DM BF
.DA TX
.EV 1
.IF !#+X-1 .SA
..
.DM EF
.BR
.EV Ø
.DI
.NR W -#V
.CH FO #W
.IF #N-#P-#W .CH FO #N+1
..
.DM SA
--------------------
.BR
..
.DM TR
.BF
.NF
.FE
.FI
.EF
..
.DM FN
.DI FE
..
.DM FT
.EV 1
.NF
.TX
.RM TX
.DI
.FI
.EV Ø
..
.AT 1 HD
.AT -#B FO
.AT -4 NM
.CH FO 7Ø
.AT -#B ~~FN~~ FT
.CH FO -#B
.EV 1
.AU 1
.LN 55
.EV Ø
```

Please remember that it is not necessary to fully understand how these macros work as long as you know how to use them.

IV. TWO COLUMN OUTPUT

The SSB processor does not support backward line feeds so it is necessary to use some operator intervention in order to produce two column output. The following set of macros will produce two column output, each column being 31 characters wide. When the text of the first column reaches the bottom of the page, the string "REPOSITION PAPER" will be output to the terminal and a "STOP" command is executed. At this time the operator should reposition the paper to the top of the page and then restart the processor by typing any key but "S".

```
.LN 31
.NR A Ø
.DM HD
.IF #A .PA
:SP 2
.AU 1
.IF !#+A-1 .TL ''title''
.IF #A-1 :SP
:SP 2
.IF #A-1 .LM 34
..
.DM FO
:SP 2
.LM Ø
.IF #A-1 .TL ''-%-''
.IF #A-1 .NR A Ø
:PG
..
.DM PA
.TM REPOSITION PAPER
:ST
.PN %-1
..
.AT 1 HD
.AT -5 FO
.BR
```

It should be noted that these macros also contain their own special set of header and footer macros which may be modified as desired.

V. FORM LETTERS

The last set of macros and examples deal with form letters. These macros are shown with some sample text and make extensive use of disk data files. This example should be thoroughly studied before trying to make use of disk data file commands. The RP (repeat) command is used so that the file is repeated over and over, until the end of file has been reached in the data file (number register E is non zero). The macro creates a name and address header at the top of each page. Following is "Dear (persons name)" and the text of the letter. The sample program is shown below, followed by the sample data file, and then a sample of the output produced by the processor.

```
.OF
.JU N
.NF
.DI NM
.RI
.BR
.DI
.IF #E .EX
.SP 6
.NM
.RI
.RI
.SP 3
.FI
Dear
.NM
.SP
.SI 5
We are writing to you to inform you that your
.RI
Insurance policy is about to expire.
Your policy number is
.RI
and expires on
.RI S
\.
If you desire renewal, please send payment by
the end of this month.
If payment is not received, your policy will be terminated.
.RI
Thank you for your attention to this matter.
.SP 2
.NF
Thank you
.SP 3
 Agent
.NB
.RP
```

The sample data file is as follows:

```
John Doe>
1313 Riverside Ave.>
Akron, Ohio 44225>
Fire>
F3-4322-946>
March 15, 1975>>
Bill Jones>
1111 Crescent Street
Apartment #12>
Kingston, New York 10011>
Automobile>E5-4936-263>March 14, 1975>
This is your second and final notice!>>
Hiram Johnson>
RR #3>
Lotson, Virginia 32004>
Life>
B1-2234-123>
March 12, 1975>>
```

As can be seen in the above sample data file, items may be placed one per line, or multiples per line as desired. The following is the output obtained from the first block of the data file.

SYSTEM ADAPTATIONS

There are three features which can be user adjusted. These are treated separately below.

I. MACRO STORAGE SPACE

The macro storage space is presently set to approximately 4K and resides at the top of the first 12K block of memory. In 99% of all applications, this space will be much more than sufficient. If more memory is available, and you are requiring more macro space, the size of this space can be expanded. The end of the space address is stored at location $Ø212 (LMACRO) and may be changed as needed.

II. DOUBLE CHARACTERS

Three commands exist in the processor which require special printer hardware. These ara double height (DH), double width (DW), and double both (DB). Some commercially available printers will print single lines of double size characters if a special control character is received prior to the line. The double height control character ($12) is stored in location $Ø21C. The double width control character ($ØE) is stored in location $Ø21D. These may be changed as required.

III. SUSPENDING EXECUTION OF THE TEXT PROCESSOR

Typing Control C during the execution of the text processor will suspend the processing and print "..BREAK.." on the terminal. Typing the letter "S" will cause the processor to exit to the operating system. Any other letter will cause processing to continue. The text processor is configured to use the MIKBUG PIA terminal interface. If you are instead using an ACIA for terminal I/O, the base address of the ACIA should be placed in locations $Ø213.

6809 SOURCE LISTING

```
0200                            ORG     $0200

                        * )))))) PROGRAM ENTRY POINT ((((((

0200 7E   0223      A START     JMP     INTRO

                        * JUMP TABLE

0203 7E   D286      A OUTCH     JMP     OUTEEE    TERMINAL CHARACTER OUTPUT
0206 7E   D289      A INCH      JMP     INEEE     TERMINAL CHARACTER INPUT
0209 7E   D283      A MON       JMP     ZWARMS    ADDRESS IN MONITOR TO EXIT T
020C 7E   D2C1      A POUCH     JMP     ZPUTCH    DIRECTED OUTPUT VECTOR

020F      23        A           FCB     EDIT      REVISION CONTROL INFO
0210      200F      A MACROS    FDB     ENDTP     MACROS CAN START AT END OF P
0212      2EFD      A LMACRO    FDB     $2EFD     LAST AVAILABLE FOR MACROS
0214      01FF      A STACK     FDB     $01FF     H/W STACK AREA

0216      3E        A ITEMCH    FCB     $3E       ")" DEFAULT ITEM CHARACTER
0217      12        A DHCHAR    FCB     $12       DOUBLE HEIGHT CONTROL CHAR
0218      0E        A DWCHAR    FCB     $0E       DOUBLE WIDTH CONTROL CHAR
0219      0F        A SWCHAR    FCB     $0F       SINGLE WIDTH CONTROL CHAR
021A      1B        A ESCHAR    FCB     $1B       ESCAPE CONTROL CHAR
```

6800 SOURCE LISTING

```
0200                            ORG     $0200

                        * )))))) PROGRAM ENTRY POINT ((((((

0200 7E   0223      A START     JMP     INTRO

                        * JUMP TABLE

0203 7E   D286      A OUTCH     JMP     OUTEEE    TERMINAL CHARACTER OUTPUT
0206 7E   D289      A INCH      JMP     INEEE     TERMINAL CHARACTER INPUT
0209 7E   D283      A MON       JMP     ZWARMS    ADDRESS IN MONITOR TO EXIT T
020C 7E   D2C1      A POUCH     JMP     ZPUTCH    DIRECTED OUTPUT VECTOR

020F      23        A           FCB     EDIT      REVISION CONTROL INFO
0210      1F16      A MACROS    FDB     ENDTP     MACROS CAN START AT END OF P
0212      2EFD      A LMACRO    FDB     $2EFD     LAST AVAILABLE FOR MACROS
0214      01FF      A STACK     FDB     $01FF     H/W STACK AREA

0216      3E        A ITEMCH    FCB     $3E       ")" DEFAULT ITEM CHARACTER
0217      12        A DHCHAR    FCB     $12       DOUBLE HEIGHT CONTROL CHAR
0218      0E        A DWCHAR    FCB     $0E       DOUBLE WIDTH CONTROL CHAR
0219      0F        A SWCHAR    FCB     $0F       SINGLE WIDTH CONTROL CHAR
021A      1B        A ESCHAR    FCB     $1B       ESCAPE CONTROL CHAR
```