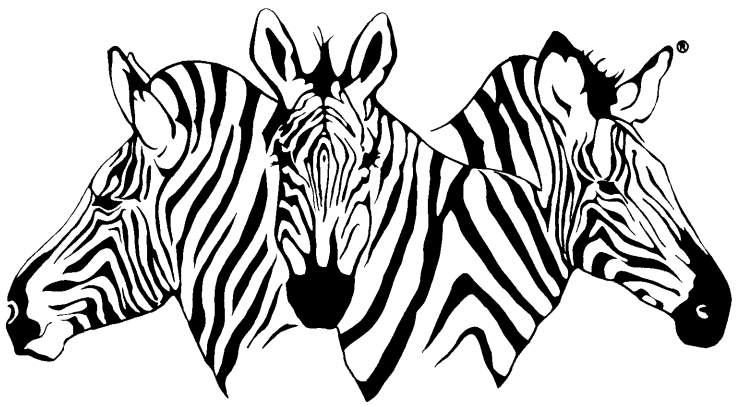
RUNOFF
reference manual

88A00781A01



ZEBRA® BY GA

RECORD OF REVISIONS

Title: RUNOFF Reference Manual

Document No. 88A00781A01

Date	Issue
March 1984	Original Issue

NOTICE

The information contained in this document is subject to change without notice.

General Automation makes no warranty or representation with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. General Automation shall not be liable for errors contained herein.

General Automation assumes no responsibility for the use or reliability of its software on equipment that is not furnished by General Automation.

RUNOFF
reference manual

88A00781A01

1045 South East Street P.O. Box 4883
Anaheim, California 92803
(714)778-4800 (800)854-6234
TWX 910-591-1695 TELEX 685-513

RECORD OF REVISIONS

Title: RUNOFF Reference Manual

Document No. 88A00781A01

Date	Issue
March 1984	Original Issue

NOTICE

The information contained in this document is subject to change without notice.

General Automation makes no warranty or representation with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. General Automation shall not be liable for errors contained herein.

General Automation assumes no responsibility for the use or reliability of its software on equipment that is not furnished by General Automation.

FOREWORD

This document is one of a family of ZEBRA reference manuals devoted to PICK processors that are on call within the PICK operating system. Before reading this document and using the processor described, it is recommended that you first become familiar with the PICK terminal control language and file structure. These subjects are thoroughly covered in 88A00782A, listed below with other documents covering PICK processors.

Document No.	Title
88A00757A	PICK Operator Guide
88A00758A	ACCU-PLOT Operator Guide
88A00759A	COMPU-SHEET Operator Guide
88A00760A	Quick Guide for the PICK Operating System
88A00774A	PICK Utilities Guide
88A00776A	PICK ACCESS Reference Manual
88A00777A	PICK SPOOLER Reference Manual
88A00778A	PICK BASIC Reference Manual
88A00779A	PICK EDITOR Reference Manual
88A00780A	PICK PROC Reference Manual
88A00782A	Introduction to PICK TCL and FILE STRUCTURE
88A00783A	PICK JET Word Processor Guide

TABLE OF CONTENTS

runoff 1

1.1 INTRODUCTION

RUNOFF is a text processor that takes text prepared with the PICK EDITOR and produces formatted output such as memos, manuals, reports, etc. RUNOFF uses commands which control justification, page titling and numbering, spacing and capitalization. These commands may be easily inserted, edited and corrected with the PICK EDITOR. New material may also be inserted or deleted, while unchanged text need not be retyped. RUNOFF also lets you combine separate textual material into a single report and insert duplicate text into different reports.

Multiple input items are treated as a single source text file. A source text item may contain a command which causes RUNOFF to CHAIN to another file item. This makes it possible to CHAIN file items together without doing a SELECT or SSELECT. Items included in the RUNOFF verb's item-list may chain to other items within the same file. When the chain ends, processing continues with the next item from the item-list.

A source text item may also contain a command which causes RUNOFF to READ a second file item and then resume processing of the first item. This makes it possible to insert the text from a single file item into the output from many other file items. The RUNOFF verb format is:

```
RUNOFF file-name item-list {(options)}
```

Option	Explanation
C	Suppresses the .CHAIN & .READ commands.
I	Outputs the name of the next item to be RUNOFF.
J	Suppresses hiliting.
N	Causes output to the terminal to be continuous; that is, RUNOFF will not pause at the bottom of a page and wait for a carriage-return.
Nnn	Sets the number of times BOLDFACE letters are overprinted.
P	May be used to direct output to the line printer.
S	Suppresses underlining and boldface when RUNOFF output is directed to a CRT.
U	Forces output to uppercase.

The following is an example of RUNOFF Source Text which calls other RUNOFF items via the READ and CHAIN commands.

```
>RUNOFF MANUAL-FILE CHAPTER1

 Item CHAPTER1:

001 .READ MANUAL-FILE I10
002 .READ MANUAL-FILE I20
003 .READ MANUAL-FILE I30
006 .CHAIN MANUAL-FILE CHAPTER2
```

NOTE: This CHAIN command does not return to the calling item and, therefore, is the last command in this item.

1.2 SOURCE FILE FORMAT

The source file contains the text that will appear on the final copy, as well as commands to specify formatting and alternate sources of input.

Each line of input source text is processed in the text mode except those beginning with a period. A line beginning with a period is assumed to be a command line and is processed in the command mode. A command line may contain one or more commands, each starting with a period. The commands provide formatting information and select various modes of operation. Examples of RUNOFF command:

```
.SK.BOX 1,78.SK

.BP.J.PARAGRAPH 0.LINE LENGTH 74.LEFT MARGIN 2

.SECTION 2 INTRODUCTION TO RUNOFF

.INDEX 'RUNOFF Introduction'

.BOX OFF.C
```

RUNOFF fills each output line by adding successive words from the source text until one more word will not fit on the line. The line is then justified by inserting blank spaces between words at random until the last word in the line exactly meets the right margin.

RUNOFF may be set to fill output lines without justifying the right margin. When filling lines, spaces and end-of-lines are treated only as word separators. Multiple word separators are stripped from the input.

RUNOFF may be set to transmit the input source text to the output without filling lines or justifying margins. In this mode, multiple spaces and end-of-lines are not stripped from the input. Some of the commands cause a BREAK in the output. A BREAK means that the current line is output without justification. This occurs at the end of paragraphs.

1.3 RUNOFF COMMANDS

RUNOFF commands are stored along with the text in the source file. These commands are distinguished from the text by a period at the start of a command line. A command line may contain one or more commands, each starting with a period. The commands provide formatting information and select various modes of operation.

NOTE: In the following descriptions of RUNOFF commands, valid command abbreviations are shown enclosed in parentheses (where such abbreviated forms of the command exist). They should not be enclosed in parentheses when used.

1.3.1 .BEGIN PAGE (.BP)

BEGIN PAGE causes a BREAK followed by an advance to a new page. The page number is incremented and the page heading (if set) is printed.

The RUNOFF processor will execute line-skips after a BEGIN PAGE command if they are explicitly requested. Instruction strings of the form, .BP.SK 3, will execute a page-eject, followed by three blank lines.

1.3.2 .BOX m,n /.BOX OFF (.BOX)

The BOX command causes the following text to be enclosed in a box with the width parameter specified by 'm' (left margin) and 'n' (right margin). The text will continue to be 'boxed' until a "BOX OFF" command is encountered. As an example:

```
001 .BOX 4,74.CENTER
002 This is an example of a BOX.
003 .BOX
```

```
!------------------------------------------------------------!
!                 This is an example of a BOX.               !
!------------------------------------------------------------!
```

1.3.3 .BREAK (.B)

BREAK causes any partially filled line to be output before processing the next input line. Does not justify broken lines.

1.3.4 .CAPITALIZE SENTENCES (.CS)

This command puts RUNOFF in the capitalize sentences mode. In this mode, the first letter of each sentence is capitalized. The first letter after a '.', '?', or '!', followed immediately by either a space or an end-of-line (Attribute Mark) is capitalized. The capitalize sentences mode also causes the following characters to be followed by a double space or an end-of-line: '.', '?', '!', ':', and ';'. CAPITALIZE SENTENCES is one of the STANDARD settings. (See STANDARD command.) Note: This command is ignored in the NOFILL (NF) mode. To cancel this command, use the NOCAPITALIZE SENTENCES (NCS) command (Section 1.3.26).

1.3.5 .CENTER (.C)

CENTER causes the next line to be input in NOFILL mode and centered on the next line of output. This command causes a BREAK to occur.

1.3.6 .CHAIN {DICT}{file-name} item-id

This command causes RUNOFF to CHAIN to the input text file item indicated. The DICT and file-name are both optional. If DICT is not specified, the data section of the file is assumed. If no file-name is given, the item will read from the same file as the item being processed.

The text input from this item is processed and output without any parameter or mode changes. RUNOFF does not resume processing text from the current source of input. This command does not cause a BREAK.

The .CHAIN command will scan the string following the command, looking for an item-id or a file name. The legal delimiter for the item-id or file-name is a blank. The file-name and item-id may have an included period. If there is more than one string following the CHAIN command which is delimited by a blank, then the next-to-the-last field will be taken to be the file-name, and that file will be opened. The last field delimited with a blank will be considered the item-id, and it will be retrieved by the RUNOFF processor to be executed next. You may include a comment statement after the CHAIN; however, the line is considered exhausted when the processor encounters an end-of-line mark, or when it encounters a period preceded by a space.

If the processor opens a file when executing a CHAIN statement, that file will be the file from which all succeeding items are retrieved, until another file is specified by another CHAIN statement.

The RUNOFF C option will suppress the .CHAIN command if it is desired to RUNOFF only one element of a chained structure. The RUNOFF I option will cause the name of the next item to be output by RUNOFF to be placed in the last line of the last item RUNOFF. To separate the item-id from the data, it is necessary to include a .B or .SK command before the .CHAIN. This is used for tracing chained sequences in large documents.

1.3.7 .CHAPTER text

This command may be used to handle automatic chapter numbering and formatting. This command has the same effect as:

```
.BEGIN PAGE.CENTER
.CHAPTER n
.SPACE 2
text
.SPACE 2
```

where the chapter number n is incremented automatically. For example:

```
.CHAPTER RUNOFF
```

would produce:

CHAPTER 1

RUNOFF

1.3.8 COMMENT INSTRUCTION (.*)

The .* command will inform the RUNOFF processor that all of the rest of the text in the line in which it occurs is a comment. It must either be at the beginning of the line, or after another command in a command line. It is always the last command in a line. This allows text to be commented on, and the intent of READs or CHAINs to be noted.

1.3.9 .CONTENTS

This command prints the table of contents accumulated by preceding CHAPTER and SECTION commands. This command should be used at the end of the RUNOFF source file. An example of the results of this command can be seen by looking at the TABLE OF CONTENTS at the beginning of this manual.

NOTE: The LINE LENGTH and LEFT MARGIN of the Table of Contents is determined by those settings that are in effect when the first .CHAPTER or .SECTION command is encountered.

1.3.10 .CRT

This command directs the RUNOFF output to the user's terminal. CRT is one of the STANDARD settings. (See the STANDARD command.)

1.3.11 .ENDCASE (.EC)

EC causes both the UC and LC conditions to be negated and allows the text to output in its natural condition. The forms ^^ and \\ cause the text to switch to uppercase and to lowercase in the same way that UC and LC cause the switch, except that ^^ and \\ may be embedded in a line. Turning off the condition ^^ or \\ requires the use of EC.

1.3.12 .FILL (.F)

Fill puts runoff into the line fill mode. Words are processed until there are enough to fill a line without overflowing it. If justification mode is on, runoff will insert spaces in the line at random to make the right margin line up. Fill is one of the standard settings. (See the STANDARD command.)

1.3.12.1 Treatment of Hyphens

Hyphens which are surrounded by alphabetic characters will allow a word-break on the hyphen in fill and justify modes. That is, if a term is a concatenation of two words separated by a hyphen, and the line overflows within the second part of the term, then the first part and the hyphen are left in the line, and the next line is commenced with the second part of the word.

Similarly, if a line in the source text terminates with a hyphen preceded by an alphabetic character, and the first character in the next line is an alphabetic character, then the last word in the line and the hyphen will be concatenated with the first word in the next line and output together in a line with the hyphen between the two parts. If there is a line overflow which occurs during this process, the hyphenated word will be handled as above. What the processor will not do is remove the hyphen.

If you wish to keep a hyphenated string of characters or words together on one line, place an underscore or back-arrow,(depending on your terminal), before each hyphen. The hyphenated string will then be considered one word, and will not be broken at a hyphen.

1.3.13 .FOOTING

FOOTING causes the next line of text and/or options that immediately follows the command to be input in nofill mode and stored in a page footing buffer. The page footing buffer will be output at the bottom of each page. The page footing may be changed with successive FOOTING commands. The general form of the FOOTING command is:

{text}{'option(s)'}{text}{'option(s)'}...

The following characters have special meaning in page footings and headings. They may be inserted anywhere in the line after the FOOTING command (before, after, or in FOOTING text) and may also appear without text. Multiple options may be strung together within single quotes. (Separation by commas is not allowed.)

Option	Explanation
'C'	Centers the line according to the page width set by the TERM command.
'D'	Prints out the Date in '01 JAN 1982' format (11 characters).
'F'	Prints out the File-Name.
'Fn'	Prints out the File-Name, left justified in a field of 'n' spaces.
'I'	Prints out the Item-id.
'In'	Prints out the Item-id, left justified in a field of 'n' spaces ('n' specified by the user).
'L'	Performs a carriage return/line-feed (CR/LF) before each footing if 'L' is first in the footing line, or after each footing if 'L' is last in the footing line.
'P'	Prints out the page number, right justified in a field of four spaces, with blank fill.
'Pn'	Prints out the page number, left justified in a field of 'n' spaces ('n' specified by the user).
'T'	Prints out the Time and Date (22 characters long).

FOOTING causes a BREAK and also is one of the STANDARD settings. (See the STANDARD command.)

1.3.14 .HEADING

HEADING causes the next line to be input in NOFILL mode and stored in a page heading buffer. The page heading buffer will be output at the top of each page.

The page heading may be changed with successive HEADING commands. The special characters described under the FOOTING command may also be used in page headings.

The HEADING command causes a BREAK and also is one of the STANDARD settings.

1.3.15 .HILITE c / .HILITE OFF

HILITE causes the character specified by 'c' to be printed out at the extreme right margin for every line of text until a HILITE OFF command is encountered.

The HILITE command does not cause a break in the text. This allows parts of paragraphs to be highlighted in justify or fill mode.

If you wish to align the HILITE command with a paragraph, it may be necessary to put the HILITE c command after the first line of filled or justified text, and to put the form .BREAK.HILITE at the end of the paragraph.

If the term .HILITE is the last character string in the command line, then it is equivalent to .HILITE OFF. The RUNOFF J option will suppress highlighting.

1.3.16 .INDENT n (.I)

INDENT causes the next line of output to be indented by n spaces to the right of the left margin. N may be negative to cause the line to begin left of the left margin. If n is missing, n=1 is assumed. This command causes a BREAK to occur.

1.3.17 .INDENT MARGIN n (.IM)

This command causes the left margin to be increased by n spaces and the line length to be decreased by n. Negative n may be used to decrease the left margin and increase the line length. This command causes a BREAK to occur.

1.3.18 .INDEX text

INDEX causes the text specified to be stored in an index list. The text may be one word, or several words enclosed in single quotes. The word, or word string, along with the current page number, are put in a sorted index list. The index can be printed by the PRINT INDEX command.

1.3.19 .INPUT

The INPUT command causes RUNOFF to read the next line of souce text from the user's terminal. The text input from the terminal is processed and output without a BREAK or mode change.

1.3.20 .JUSTIFY (.J)

JUSTIFY puts RUNOFF in the FILL and JUSTIFY mode. RUNOFF fills each output line by adding successive words from the source text until one more word will not fit on the line. The line is then justified by inserting blank spaces between words at random until the last word in the line exactly meets the right margin. JUSTIFY is one of the STANDARD settings.

1.3.21 .LEFT MARGIN n

This command sets the left margin to n spaces. If n plus the current line length exceeds the maximum line length, this command is ignored. A LEFT MARGIN of 0 is one of the STANDARD settings.

1.3.22 .LINE LENGTH n

This command sets the line length to n characters (not counting the left margin). If n plus the current left margin exceeds the maximum line length, this command is ignored. A LINE LENGTH of 60 is one of the STANDARD settings.

1.3.23 .LOWER CASE (.LC)

This command puts RUNOFF into lowercase mode. In lowercase mode, all letters are automatically made lowercase. They may then be changed to uppercase by various text commands or control characters. (See Section 1.4, SPECIAL CONTROL CHARACTERS.)

1.3.24 .LPTR

This command directs the RUNOFF output to the line printer.

1.3.25 .NOCAPITALIZE SENTENCES (.NCS)

This command resets the CAPITALIZE SENTENCES mode.

1.3.26 .NOFILL (.NF)

This command resets both JUSTIFY and FILL modes. Input text lines will be output as they are, (after possible elimination of special control characters) without removal of extra spaces. Output lines will not be filled nor will right margins be justified. This command causes a BREAK.

1.3.27 .NOJUSTIFY (.NJ)

This command resets the JUSTIFY mode.

1.3.28 .NOPAGING (.N)

The N option may be used to eliminate the wait for terminal input at the end of each page printed on the terminal.

1.3.29 .NOPARAGRAPH

This command resets the paragraph mode. Blank input text lines and spaces at the beginning of a line will be ignored in justify mode.

1.3.30 .PAGE NUMBER n

This command sets the current page number to n. If n is missing, n=1 is assumed.

1.3.31 .PAPER LENGTH n

This command sets the paper length to n lines.

1.3.32 .PARAGRAPH n (.P)

This command causes any blank line or any line which starts with a space to be considered as the start of a new paragraph. This allows normally typed text to be justified without any special commands. n sets the number of spaces paragraphs are to be indented or unindented. A paragraph causes a BREAK followed by (line spacing +1)/2 blank lines. The PARAGRAPH command may be set to a negative number. An illustration of the use of a negative paragraph setting to decrease the left margin is shown below:

```
001 .SK.PARAGRAPH -4 .LEFT MARGIN 13 .LINE LENGTH 63
002  1. The user enters the command "Z" to the DEBUGGER prompt character
003 "*". The DEBUGGER responds with "PROG NAME?", the user enters the
004 program name. This allows the DEBUGGER access to the symbol table
005 created during compilation. Alternatively, if the user uses the
006 "(D)" during run time, access to the symbol table is already
007 established, and use of the "Z" command is unnecessary.
008  2.  To find out how far in the loop the program progressed, the
009 user looks at the variable "I" by entering "/I". The DEBUGGER
010 responds with
011 "11 =", at which the user may change the value of "I" if desired.
012 The user may then want to look at all of the values in the array by
013 entering "/ARRAY". The DEBUGGER responds with "ARRAY(1)=1=", the
014 user presses
015 return and the DEBUGGER continues with the next "array slot"
016 (i.e., "ARRAY(2 etc.)=2="). Once "ARRAY(10)=10=" has been reached
017 the . . . Etc.
018 .PARAGRAPH 0 .LEFT MARGIN 2 .LINE LENGTH 74
```

The text lines beginning with 1. and 2. are preceded by one space to indicate paragraphing. The -4 produces a hanging indent of 4 spaces. The left margin and line length commands are adjusted to indent the left and right margins accordingly. The above source text would print:

1. The user enters the command "Z" to the DEBUGGER prompt character "*". The DEBUGGER responds with "PROG NAME?", the user enters the program name. This allows the DEBUGGER access to the symbol table created during compilation. Alternatively, if the user uses the debug option "(D)" during run time, access to the symbol table is already established, and use of the "Z" command is unnecessary.
2. To find out how far in the loop the program progressed, the user looks at the variable "I" by entering "/I". The DEBUGGER responds with "11 =", at which the user may change the value of "I" if desired. The user may then want to look at all of the values in the array by entering "/ARRAY". The DEBUGGER responds with "ARRAY(1)=(1)=", the user presses return and the DEBUGGER continues with the next "array slot" (i.e., "ARRAY(2)=2=" etc.). Once "ARRAY(10)=10=" has been reached the ... Etc.

1.3.33 .PFILE n

This command controls the Spooler print file allocation with 'n' not greater than 60 (the limit the spooler can manage in an open state at any one time). For example, this command might be used in conjunction with the .CONTENTS command with .PFILE n preceding that instruction. This would cause the table of contents to be a separate Spooler entry. If used with "SP-ASSIGN HS" you may print your table of contents followed by text body rather than at the end.

1.3.34 .PRINT

The PRINT command causes RUNOFF to print the next line of input text on the user's terminal.

1.3.35 .PRINT INDEX

This command causes the sorted index list of words and page numbers to be printed. The index is sorted into alphabetical order and printed in two columns per page. Note: This command changes the tab settings, and causes a BEGIN PAGE command to be performed.

1.3.36 .READ {DICT}{file-name} item-id

This command causes RUNOFF to read the file item indicated. The DICT and file-name are both optional. If DICT is not specified, the data section of the file will be used. If no file-name is given, the item will be read from the same file as the item being processed. The text input from this item is processed and output without any parameter or mode changes. After processing this item, RUNOFF resumes input with the next line of the current source of input. This command does not cause a BREAK.

The .READ command will scan the string following the command, looking for an item-id or a file name. The legal delimiter for the item-id or file-name is a blank. The file-name and item-id may have an included period. If there is more than one string following the READ command which is delimited by a blank, then the next-to-the-last field will be taken to be the file-name, and that file will be opened. The last field delimited with a blank will be considered the item-id, and it will be retrieved by the RUNOFF processor to be executed next. If the statement is a READ, then the processor will eventually return to this item and continue processing it. When it does, it will commence at the beginning of the next line in the item. Therefore, no statements which occur after the READ statement in the line will be executed. You can include a comment statement after the READ however. Therefore, for the purposes of the READ command, the line is considered exhausted when the processor encounters an end-of-line mark, or when it encounters a period preceded by a space.

The C option will suppress the .READ command if it is desired to RUNOFF one element of a chained or treed structure. The I option will cause the name of the next item to be output by RUNOFF to be placed in the last line of the last item RUNOFF. This is most useful with large documents.

1.3.37 .READNEXT

This command is used to read data from a preselected LIST. It has an effect only if, prior to entering RUNOFF, a SELECT, SSELECT, QSELECT or GET-LIST statement has been entered, which selects a list of values. Each READNEXT command in RUNOFF will extract one value from the select-list and place it in the text stream. READNEXT does not cause a break. If there is no preselected list, or when the list is exhausted, the READNEXT command will cause a termination of RUNOFF, and a return to TCL.

This command is particularly useful when form-letters are to be generated. For example, it may be necessary to insert the name and address of each recipient of the letter from a separate file. A SSELECT statement is used to extract the relevant data from the file and save it in a list. A series of READNEXT statements will insert the data into the text of the letter. At the end of the letter, a CHAIN statement may be used to restart the next letter. When the list is exhausted, the RUNOFF will stop. The commands necessary to generate a form letter are:

```
{S}SELECT file-name {selection-criteria} attribute-list

.READNEXT

.CHAIN item-name
```

The selected attribute-list contains all the variable information to be 'written' into the form letter. The use of '.READNEXT' commands reads each of these variables and causes them to be 'written' into the letter. The '.CHAIN' command causes the letter to be repeated so long as there is variable information in the selected attribute list. The following example demonstrates the generation of a form letter.

Assume the dictionary of the accounts payable file for a company contains the following three Attribute defining items:

Name	Account	Amount
001 A	001 A	001 A
002 1	002 2	003 3
003 CUSTOMER NAME	003 ACCOUNT TYPE	003 AMOUNT DUE
004	004	004
005	005	005
006	006	006
007	007	007
008 A1:","	008	008 A;3(MR2$,):"."
009 L	009 L	009 R
010 25	010 30	010 10

The dictionary also contains the following form letter written in RUNOFF:

```
 LETTER

001 .SK 8
002 Dear Mr.
003 .READNEXT
004  Our records show that your
005 .READNEXT
006 account is overdrawn by the amount of
007 .READNEXT
008 We would appreciate prompt payment.
009  Thank you,
010            John Dunne
011 .SK 2
012    President CELEBRITY SERVICES CO.
013 .SK 3
014 .BP
015 .CHAIN LETTER
```

The data file contains items such as the following three:

```
 250                    251                 252
 001 Jim Smith          001 Andy Jones      001 Rob Reedy
 002 Basketball Shoes   002 Guitar String   002 Voice Lesson
 003 25000              003 12345           003 452359
```

To generate the form letter, the data file is first sort selected by the name with the attribute list of NAME ACCOUNT and AMOUNT:

```
SSELECT ACC-PAYABLE BY NAME NAME ACCOUNT AMOUNT
```

The command will generate a selected list containing the following information:

```
001 Andy Jones,
002 Guitar String
003 $123.45.
004 Rob Reedy,
005 Voice Lesson
006 $4,523.59.
007 Jim Smith,
008 Basketball Shoes
009 $250.00.
```

Note that the correlatives on the names and on the amounts have been performed. Now, by issuing the following RUNOFF command, the form letters are generated:

```
RUNOFF DICT ACC-PAYABLE LETTER (P)
```

The form letters will be printed as follows:

Dear Mr. Andy Jones,

Our records show that your Guitar String account is overdrawn by the amount of $123.45. We would appreciate prompt payment.

Thank You,

John Dunne

President CELEBRITY SERVICES CO.

(Next page)

Dear Mr. Rob Reedy,

Our records show that your Voice Lesson account is overdrawn by the amount of $4,523.59. We would appreciate prompt payment.

Thank You,

John Dunne

President CELEBRITY SERVICES CO.

(Next page)

Dear Mr. Jim Smith,

Our records show that your Basketball Shoes account is overdrawn by the amount of $250.00. We would appreciate prompt payment.

Thank You,

John Dunne

President CELEBRITY SERVICES CO.

1.3.38 .SAVE INDEX file-name

This command causes chapter and page number information of indexed data in a text to be saved in a separate file. Each indexed word (or string) is stored as an individual item using the word as the item-id, the chapter (where that word is referenced) as the first attribute, and the page number as the second attribute. Multiple values are stored in these attributes as multiple references to the same indexed word are encountered. The resulting file may then be operated on by the ACCESS processor to generate listings for the chapter and page number information of all indexed words in a text.

The 'file-name' is the name of the file in which the chapter and page information is to be stored. This must be a separate file from the text file or the data in the text file will be destroyed. Also, the .SAVE INDEX command is placed in the text item itself and must precede the '.INDEX' commands. Since only indexed data which has preceded the '.SAVE INDEX' command will be saved in the specified file, it is usually placed at the end of the file.

1.3.39 .SECTION n Text

This command may be used in conjunction with the CHAPTER command to handle automatic chapter section numbering and formatting. The SECTION command automatically starts the next section at section number level n, where n is the range 1-5. The text is printed following the section number and SKIP occurs. The text is recorded as the section heading in the TABLE OF CONTENTS. If no text appears on the SECTION command, the section is not recorded in the TABLE OF CONTENTS. Section numbers are incremented automatically and the section number is printed in the form i.j.k.l.m with n digits printed.

Conventionally, the .SECTION command is followed by a blank line before the next paragraph starts. Since the SECTION command causes a break which terminates the preceding paragraph, and since the text following the SECTION command is placed immediately into an output line and output prior to a consideration of the next line, the blank line after the SECTION command can be avoided by not indenting the first line of the next paragraph. That is, if the processor does not know that the next line starts a paragraph, it will not skip a line. It may be necessary to use an INDENT MARGIN if paragraph indentation is desired, however.

1.3.40 .SET TABS n,n,n, ...

This command clears previous tab stops and sets new tab stops as indicated by the numeric tab positions. The tab stops (up to 30) must be greater than zero and in increasing order. They indicate tab stop positions relative to the left margin. Tabs are only in effect in NOFILL mode. The left-tab character (<) causes the next word to start at the next tab position. The right-tab character (>) causes the next word to end at the next tab position. If a tab character appears at a point in the line where no further tab stops have been set, the tab character is ignored. (See TAB SETTINGS under RUNOFF.)

1.3.41 .SKIP n (.SK)

The SKIP causes a BREAK after which n*(SPACING n) lines are left blank. If the skip would advance past the end of the page, the output is advanced to the top of the next page. If n is missing, n=1 is assumed.

1.3.42 .SPACE n (.SP)

This command has the same effect as SKIP, except that n (rather than n*(SPACING n) lines are left blank. SPACE is used where space is to be left independent of the line spacing; SKIP is used where space should be relative to the SPACING command. If n is missing, n=1 is assumed.

1.3.43 .SPACING n

This command sets the line spacing to n. The command .SPACING 2 may be used for double spacing.

1.3.44 .STANDARD

This command sets the standard (default) parameters and modes. The STANDARD command is equivalent to the following commands:

```
.CS.F.J.EC.LEFT MARGIN 0.CRT.HEADING
.FOOTING
.PARAGRAPH 0.LINE LENGTH 60
```

1.3.45 .TEST PAGE n

This command causes a BREAK followed by an advance to a new page when there are less than n lines remaining on the current page. If there are n or more lines remaining on the current page, this command has no effect. This command should be used to ensure that the following n lines are all output on the same page.

1.3.46 .UPPER CASE (.UC)

This command puts RUNOFF into uppercase mode. Alphabetic letters will be processed as they are, unless modified by special commands to control characters. This command allows users of terminals with upper and lowercase to generate the input text file without special commands to control characters. UC is one of the STANDARD settings.

The 'U' option of the RUNOFF verb also may be used to force the whole RUNOFF output to uppercase.

1.4 SPECIAL CONTROL CHARACTERS (^, \ ,^^ ,\\, &, <, >, @, ~)

RUNOFF features special control characters for 1) Upper/Lowercase, 2) Underlining, 3) Tab Setting, 4) Boldface Printing, and 5) Special Character Override.

1.4.1 UPPER/LOWERCASE

The up-arrow, ^ and back-slash, \ may be used to specify case information when preparing source text on a device without lowercase.

The up-arrow (^) causes the letter immediately following to be changed to uppercase.

```
.LC
^SPECIAL ^ CONTROL ^ CHARACTERS ARE NEEDED ...
```

This example of RUNOFF source would print as:

```
Special Control Characters are needed ...
```

Two consecutive up-arrows (^^) put RUNOFF in uppercase mode until two consecutive back-slashes (\\) are encountered.

```
.LC
^^SPECIAL CONTROL CHARACTERS \\ ARE NEEDED ...
```

This example of RUNOFF source would print as:

```
SPECIAL CONTROL CHARACTERS are needed ...
```

The back-slash (\) causes the letter immediately following to be changed to lowercase.

```
.UC
ABC M\F\G. C\O.
```

This example of RUNOFF source would print as:

```
ABC Mfg. Co.
```

1.4.2 UNDERLINING

The ampersand (&) may be used to indicate underlining. The ampersand causes the letter immediately following to be underlined.

```
.LC
THE LETTER &A IS FIRST IN THE ALPHABET
```

This example of RUNOFF source would print as:

the letter a is first in the alphabet

An ampersand may be used in conjunction with the up-arrow and back-slash to underline a series of characters. An ampersand followed immediately by an up-arrow (&^) puts RUNOFF in the underline mode until an ampersand followed immediately by a back-slash (&\) is encountered.

```
.UC
&^SPECIAL CONTROL CHARACTERS&\ ARE NEEDED ...
```

This example of RUNOFF source would print as:

SPECIAL CONTROL CHARACTERS ARE NEEDED ...

1.4.3 TAB SETTINGS

The less-than (<) and greater-than (>) characters may be used for tabbing. The left-tab character (<) causes the next word to start at the next tab position as set by the .SET TABS command. The right-tab character (>) causes only the next word to end at the next tab position.

```
.NF
.SET TABS 5,8,25
.SK 1
><&^NAME<CONVENTIONAL DATA PROCESSING NAME&\
.SK 1
>1.<Item<Record
>1a.<Attribute<Field
>1b.<Item-id<Record Key
```

This example of RUNOFF source would print as:

	NAME	CONVENTIONAL DATA PROCESSING NAME
1.	Item	Record
1a.	Attribute	Field
1b.	Item-id	Record Key

NOTE: Tab characters are only in effect in the NOFILL (NF) mode.

1.4.4 BOLDFACE PRINTING

The at sign (@) may be used to indicate **BOLDFACE** type. An at sign followed immediately by an up-arrow (@^) puts RUNOFF in the boldface mode until an at sign followed immediately by a back-slash (@\) is encountered. The number of times the boldface letters are overprinted may be set by using the numeric option of the RUNOFF verb.

```
.UC
@^SPECIAL CONTROL CHARACTERS@\ ARE NEEDED ...
```

This example of RUNOFF source would print as:

SPECIAL CONTROL CHARACTERS ARE NEEDED ...

1.4.5 SPECIAL CHARACTER OVERRIDE

The underscore or back-arrow (<-) (depending on your terminal) may be used to quote one of the special control characters or blanks. The character immediately following the underscore or back-arrow is transmitted to the output without special processing.

```
<-^SPECIAL <-^CONTROL <-^CHARACTERS ARE NEEDED ...
```

This example of RUNOFF source would print as:

```
^SPECIAL ^ CONTROL ^ CHARACTERS ARE NEEDED ...
```