M T S

The Michigan Terminal System

Volume 15: FORMAT and TEXT360

April 1977

Updated February 1979 (Update 1)
Updated January 1983 (Update 2)
| Updated March 1988 (Update 3)

The University of Michigan Computing Center
Ann Arbor, Michigan

DISCLAIMER

The MTS Manual is intended to represent the current state of the Michigan Terminal System (MTS), but because the system is constantly being developed, extended, and refined, sections of this volume will become obsolete. The user should refer to the U-M Computing News, Computing Center Memos, and future updates to this volume for the latest information about changes to MTS.

PREFACE

The software developed by the Computing Center staff for the operation of the high-speed processor computer can be described as a multiprogramming supervisor that handles a number of resident, reentrant programs. Among them is a large subsystem, called MTS (Michigan Terminal System), for command interpretation, execution control, file management, and accounting maintenance. Most users interact with the computer's resources through MTS.

The MTS Manual is a series of volumes that describe in detail the facilities provided by the Michigan Terminal System. Administrative policies of the Computing Center and the physical facilities provided are described in other publications.

The MTS volumes now in print are listed below. The date indicates the most recent edition of each volume; however, since volumes are periodically updated, users should check the file *CCPUBLICATIONS, or watch for announcements in the U-M Computing News, to ensure that their MTS volumes are fully up to date.

Volume 1: The Michigan Terminal System, January 1984
Volume 2: Public File Descriptions, January 1987
Volume 3: System Subroutine Descriptions, April 1981
Volume 4: Terminals and Networks in MTS, March 1984
Volume 5: System Services, May 1983
Volume 6: FORTRAN in MTS, October 1983
Volume 7: PL/I in MTS, September 1982
Volume 8: LISP and SLIP in MTS, June 1976
Volume 9: SNOBOL4 in MTS, September 1975
Volume 10: BASIC in MTS, December 1980
Volume 11: Plot Description System, August 1978
Volume 12: PIL/2 in MTS, December 1974
Volume 13: The Symbolic Debugging System, September 1985
Volume 14: 360/370 Assemblers in MTS, May 1983
Volume 15: FORMAT and TEXT360, April 1977
Volume 16: ALGOL W in MTS, September 1980
Volume 17: Integrated Graphics System, December 1980
Volume 18: The MTS File Editor, February 1988
Volume 19: Tapes and Floppy Disks, November 1986
Volume 20: Pascal in MTS, December 1985
Volume 21: MTS Command Extensions and Macros, April 1986
Volume 22: Utilisp in MTS, February 1988
Volume 23: Messaging and Conferencing in MTS, March 1987

Other volumes are in preparation. The numerical order of the volumes does not necessarily reflect the chronological order of their

appearance; however, in general, the higher the number, the more specialized the volume. Volume 1, for example, introduces the user to MTS and describes in general the MTS operating system, while Volume 10 deals exclusively with BASIC.

The attempt to make each volume complete in itself and reasonably independent of others in the series naturally results in a certain amount of repetition. Public file descriptions, for example, may appear in more than one volume. However, this arrangement permits the user to buy only those volumes that serve his or her immediate needs.

Richard A. Salisbury,

General Editor

April 1977

PREFACE TO VOLUME 15

Volume 15 contains the descriptions for the FORMAT and TEXT360 text-processing programs. The separate Computing Center publication entitled Introduction to FORMAT provides a rudimentary introduction to text-processing with FORMAT. The section "FORMAT" in this volume contains the complete program description for FORMAT.

Acknowledgments for the descriptions contained in this volume are as follows:

> The section "TEXT360" is reprinted with permission from the IBM publication, IBM System/360 TEXT360 Introduction and Reference Manual, form number C35-0002. This publication is no longer available from IBM.

The remainder of the descriptions in this volume were either produced or extensively modified from other documentation by the editorial and programming staff at the University of Michigan Computing Center.

April 1977

Contents

April 1977

FORMAT

INTRODUCTION

FORMAT is a text-processing language that takes the text of a document in free format, and reformats it into lines, paragraphs, and pages according to control information supplied with the text. FORMAT numbers pages, generates titles and footers, justifies text (aligns right and left margins), indents sections of text, prints text in multiple columns, etc. An index and table of contents may be generated for the document.

Control information is specified by the user in three ways:

(1) control phrases (global control)
(2) command operands (local control)
(3) special operators (immediate control)

Control phrases affect the global layout of the document. For instance, control phrases are used to specify the number of columns of text, the spacing between paragraphs, and the number of lines on a page. Control phrases appear in groups--one group at the beginning of the input text and possibly other groups later. Command operands produce a local effect on the formatting of a document; they appear in the input text. For instance, there are command operands to underscore or capitalize text, to start new paragraphs, and to begin indenting text. Special operators appear in the input text and produce an immediate effect. There are special operators to capitalize or underscore a single letter, and to print special characters such as superscripts. The three types of control information are explained in detail in later sections.

Input to FORMAT

The input for a document always begins with one or more control phrases. Each control phrase must appear on a separate input line. The last control phrase (GO) is the one that says "this is the end of the control phrases."

The GO control phrase is followed by the text of the document, containing command operands and special operators as necessary. The basic unit of text is a word, which is a string of characters delimited by blanks. Any EBCDIC character may appear in a word (certain characters are treated as special operators and command operand delimiters, but these characters may be entered indirectly). Text input is

free format--the last word on one line is immediately followed by the first word on the next line. A blank is inserted at the end of each input line as it is read, so the last word on a line need not be followed by a blank. Words are separated by one or more blanks. All blanks after the first are ignored; they are not inserted directly into the output text (however, extra blanks may be inserted into the output text to justify the line). Text is formatted into output lines without regard to the input line boundaries. Additional control phrases may appear after the beginning of the text. A command operand is used to separate them from the text.

Command operands are delimited by slashes (alternate delimiters are discussed later). For example, /P/ means begin a new paragraph. Most operands are single letters. As many operands as desired may appear between a pair of slashes; /LP/ means skip to the next line then begin a new paragraph.

Both control phrases and command operands may be entered in either upper- or lowercase. In addition, many control phrases may be abbreviated. However, for ease of reading, the examples in this description use only the full uppercase form of both control phrases and command operands.

In the following simple example, the first four lines are the control phrases; the remainder is the text of the document. Anything between a pair of slashes (/) is treated as a command operand; for instance, the first instance of /M@/ begins centering and capitalization of the following text, and the second instance terminates this. The underscores appearing in the fourth line from the bottom are special operators; each underscores the preceding character. /P/ starts a new paragraph. Notice that the actual spacing in the input text is ignored.

```
WIDTH 50
SEPARATION 1
LOWERCASE
GO
/M@/the cheshire puss/@ML2P/ "Would you tell me, please
which way I ought to go from here?" /P/"That depends a
good deal on where you want to get to," said the Cat.
/P/"I don't much care where--" said Alice./P/"Then it
doesn't matter which way you go,"        said the Cat.
/P/"--so long        as I get /U/somewhere/U/," Alice
       added as an explanation. /P/"Oh, you're sure to
do that," said the Cat, "if you only walk long enough."
/P/Alice felt that this could not be denied, so
she        tried another question. "What sort of
people live about here?"
/P/"In t_h_a_t_ direction," the Cat said, waving its
right paw round, "lives a Hatter: and in /U/that/U/
direction," waving the other paw, "lives a March Hare.
Visit either you like: they're both mad."
```

The output produced is:

THE CHESHIRE PUSS

"Would you tell me, please which way I ought to go from here?"

"That depends a good deal on where you want to get to," said the Cat.

"I don't much care where--" said Alice.

"Then it doesn't matter which way you go," said the Cat.

"--so long as I get somewhere," Alice added as an explanation.

"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

Alice felt that this could not be denied, so she tried another question. "What sort of people live about here?"

"In that direction," the Cat said, waving its right paw round, "lives a Hatter: and in that direction," waving the other paw, "lives a March Hare. Visit either you like: they're both mad."

Running FORMAT

FORMAT is invoked by the $RUN command. The logical I/O units referenced are:

SCARDS - text input to FORMAT.
GUSER - text input via the INPUT function.
SPRINT - printed output produced by FORMAT.
SERCOM - error messages (if the ERRORS control phrase is specified).

A single control phrase may be specified in the PAR field of the $RUN command. If one is specified, it is the first control phrase processed. Output from FORMAT uses the TN character set (unless the TRAIN control phrase specifies otherwise). Therefore, if output is destined for a printer, PRINT=TN should be specified on the appropriate MTS command.

The cost of running FORMAT varies with the complexity of the document input source. Small documents have a slightly higher per-page cost than large documents due to the cost of loading and initializing FORMAT. The "Cheshire Puss" example above will cost about 10 cents to produce, the major portion of this cost due to loading and initializing FORMAT. Documents that make heavy use of macros and functions will cost more than those documents that make little use of these facilities. In general, the cost of a medium size to large single-spaced document (in excess of 50 pages) will run about 3.25 to 4 cents per page, including the cost of printing; this estimate is based on March 1977 normal priority, University-Government rates.

How FORMAT Processes Text

The purpose of this section is to provide general description of how FORMAT processes text.

FORMAT has two "modes" of operation: control phrase mode and text mode. In control phrase mode, control phrases are read and stored for later use in specifying the global appearance of the document. In text mode, input text is read, formatted, and written to the output device. Simplified, the steps in text mode are:

(1) A line of input text is read into the input buffer and translated into lowercase (the LOWERCASE control phrase may be used to suppress the translation).

(2) The next (or first) word in the input buffer is examined. If the word consists entirely of a text word, the text word and a trailing blank are inserted into the output buffer in which the next output line is being constructed. If the word consists of one or more command operands, the command operands are executed, one at a time, in the order of occurrence. If the word consists of a special operator, the text which results from that operator is inserted into the buffer, followed by a blank. If the word is a mixture of the above, text pieces of the word are placed in the buffer, and command operands and special operators are processed as they are encountered. A blank is inserted in the buffer only at the end of the word.

(3) Step 2 is repeated until the input line is exhausted or the next word will not fit in the output line being constructed. If the input line is exhausted, another line is read as in step 1 and output processing continues.

(4) When the next word will not fit on the output line, the line is prepared for actual output. Extra blanks are inserted into the line if necessary for justification. Any internal codes that may have been used in the line are replaced with the proper printing characters. The line is then written to SPRINT. This step is also executed when a command operand specifies a skip to a new line. If the line written was the last line on the current page, a new page is started, with the printing of the title and page number (if any). Finally, the output buffer is emptied and step 2 is repeated.

Notice that FORMAT writes out the lines as it is reading text--it does not format the entire document and then print it out. This means that control phrases affect only the text which occurs after them.

April 1977

FORMATTING DOCUMENTS

Page Layout

Basic page layout is accomplished by specifying displacements from the upper-left corner of the page in units of lines and columns (characters). The first line of a page is line 1; the first column or print position is column position 1.

The left margin and the line of each page on which text starts is defined by the control phrase

TEXT f p

where "f" is the number of the line on which to start, and "p" is the column position. The defaults are f=5 and p=5. The space above line "f" may be used for titles and page numbers (see below and the next section).

The length of each text line is defined by the WIDTH control phrase. Assuming for now single-column text, the format is

WIDTH w

where "w" is the number of characters of text that may appear on a line, i.e., the number of column positions available for text. In effect, "w" establishes the right margin for justification of text. The default value is w=64.

The number of lines per page is defined by the control phrase

LINES l

This is the total number of lines on the page, including headers, footers, and the body of the text. The body of the text can occupy all "l" lines only if TEXT 1 has been specified and there are no footnotes or footers. "Straight text" starts on line "f", and continues as close to line "l" as the footer and footnotes permit (if there are none, the last line of text is line "l"). The default for "l" is 60.

The default values for the above control phrases are chosen with an 8-1/2 x 11 page in mind.

Figure 1 illustrates the above control phrase parameters. The outside rectangle is the printer page; the lined area is text. Straight text with no titles, footers, or footnotes is assumed for simplicity. The following trivial example may help to clarify this; if the following control phrases are given,

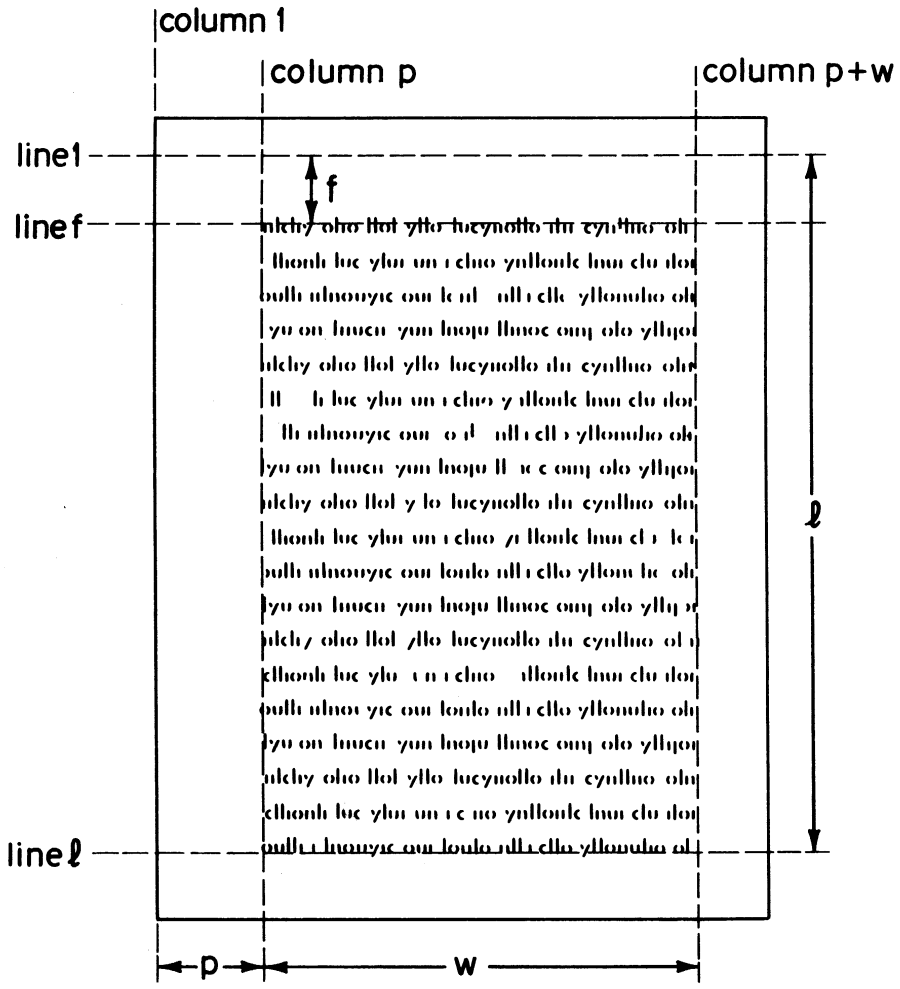


Figure 1.

April 1977

```
                        LINES 9
                        TEXT 4
                        WIDTH 40
```

a page of straight text will appear as:

```
          column position 5     column position 44
        | |                                     | | <--line 1
        | |                                     | |
        | V                                     V |
        | It was the best of  times,  it  was  the | <--line 4
        | worst  of  times,  it  was  the  age  of |
        | wisdom, it was the age  of  foolishness, |
        | it  was  the epoch of belief, it was the |
        | epoch of incredulity, it was the  season |
        | of Light, it was the season of Darkness, | <--line 9
        |                                          |
```

The WIDTH, COLUMNS, and BETWEEN control phrases are used to obtain multiple-column text. WIDTH is used to set the line length for each column; the format is

```
WIDTH w1 w2 w3 w4 w5 w6
```

where "wi" is the width of the i-th column. If fewer than 6 columns are desired, the remaining w's may be omitted (six is the maximum allowed). Each column must be at least 10 column positions wide, and the total page width, including left margin "p" and gutters between columns must not exceed 255 column positions. See Figure 2.

The size of the gutters is set by the control phrase

```
BETWEEN g1 g2 g3 g4 g5
```

where "gi" is the number of column positions between columns "i" and "i+1". If w1,...,wi are specified, then only g1,...,g(i-1) need be given (gi,...,g5 are ignored). The default value for each "gi" is 2. The spacing specified in a BETWEEN control phrase becomes effective with the next WIDTH or COLUMNS control phrase. The COLUMNS phrase may be used to start multiple-column text without specifying the exact width of each column. In this case, the format is

```
COLUMNS c
```

where "c" is the number of columns desired. The current values of "w" and g1,...,g(c-1) are used to divide the "w" column positions into "c" equal-width columns with the i-th gutter of width at least "gi". If c=1, one column of width "w" is used. As before, each column must be at least 10 column positions wide. Figure 2 illustrates the above control phrases. The outside rectangle is the printer page; the shaded area is text. For example, using the control phrases

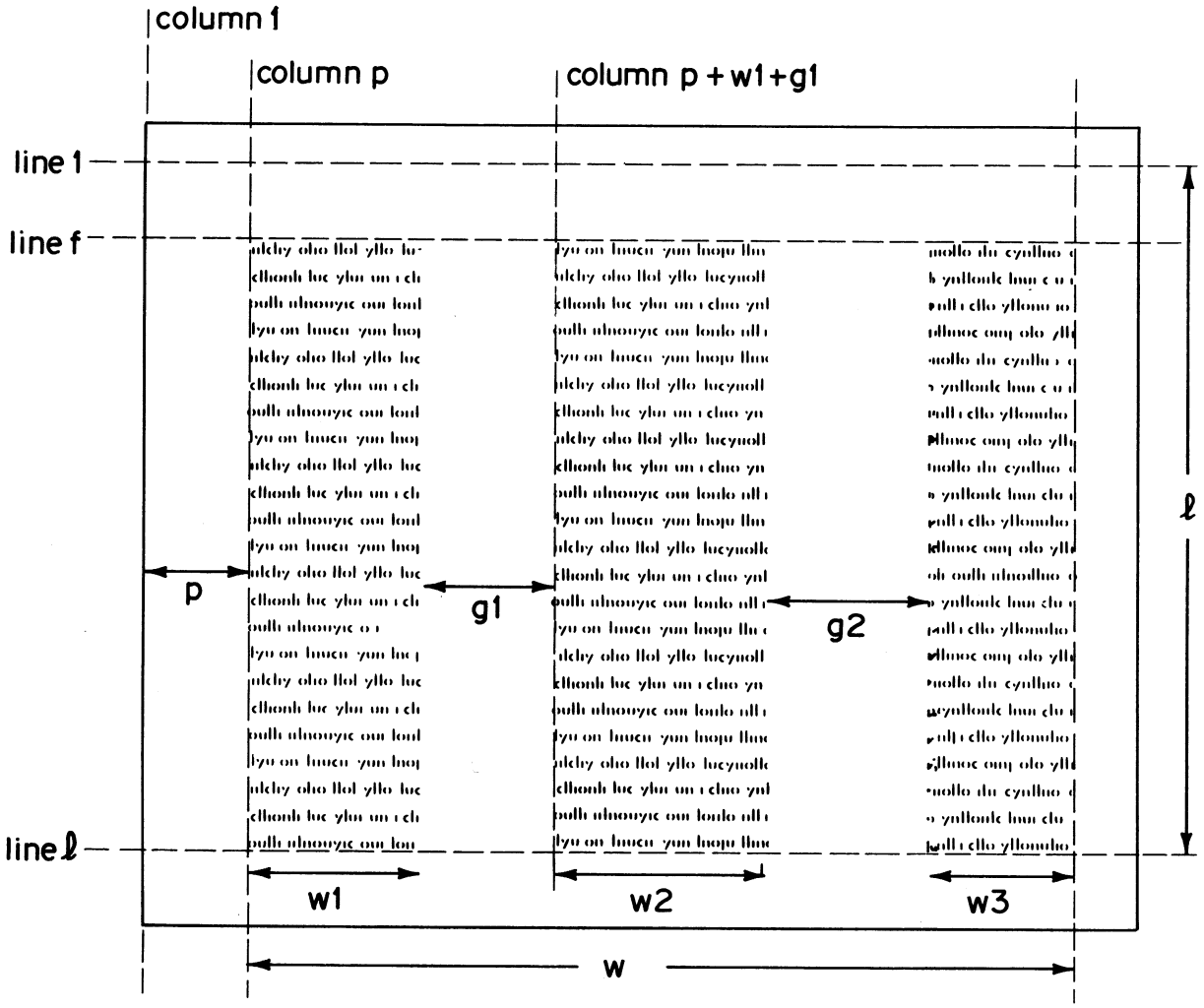

column 1
column p
column p + w1+g1
line 1
line f
line ℓ
p
g1
g2
ℓ
w1
w2
w3
w

Figure 2.

```
LINES 9
BETWEEN 6 4
WIDTH 20 15 18
```

a page of straight text might look like

```
|                                                      1 |
|                                                        |
|                                                        |
|                                                        |
| Gallia est omnis      tiam, qui ipso-   lingua, institu- |
| divisa in partes      rum lingua        tis, legibus inter |
| tres, quarum unam     Celtae, nostra    se differunt. |
| incolunt Belgae, al-  Galli appella-    Gallos ab Aqui- |
| iam Aquitani, ter-    ntur. Hi omnes    tanis Garumna |
|                                                        |
```

Headings, Footers, and Page Numbers

The control phrases covered in this section determine the presence and position of titles, footers, and page numbers. They are:

CYCLE	LTITLE	RIGHT
FOOTER	PAGE	RTITLE
LEFT	REPEAT	SUBTITLE
LFOOTER	RFOOTER	TITLE

Documents may be formatted so that titles, footers, and page numbers appear in the same position on every page, or so that righthand and lefthand pages are used. Uniformly formatted pages are described first.

By default, each page is numbered in the upper-right corner (aligned with the right margin) beginning with page number 1. This may be changed by the control phrase

```
PAGE pn pl pc
```

where "pn" is the starting page number to use, "pl" is the line of the page on which page numbers are to be placed, and "pc" is the number of column positions that the number is to be moved leftward from the right margin. The default is pn=1, pl=1, and pc=0. The page number is printed where specified even if a title or line of text occupies that position. Thus, if the page number is to be placed on the last line of the page, and text appears on that line (normally it will, if there is no footer), the page number is overprinted on the text. (The solution to this problem is to use a blank footer.) If "pl" is larger than the number of lines on the page (specified by the LINES phrase), the page number is not printed. Roman numerals and other variations are available through the PAGE function (see the section "FORMAT Functions" below). The NO PAGE control phrase stops page numbering.

The contents and position of the title are defined by the control phrase sequence

```
TITLE tl tc
text of title /E/
```

where "tl" is the line number for the title and "tc" is the column position in which it is to begin. The defaults are tl=1 and tc=1. The line(s) following the control phrase contain the title; the title must be terminated by the /E/ command operand (see the section "Command Operands"). Using the default value of "tc" aligns the title with the left margin. One method to center the title is to use

```
TITLE
/M/ text of title /ME/
```

Command operands may be used in titles; a title is not restricted to a single line. Normally, the title appears on every page. If the NO REPEAT control phrase is used, the title is printed only on the first page following the TITLE control phrase. The printing of the title may be resumed by specifying the REPEAT control phrase.

A subtitle may be defined by the control phrase sequence

```
SUBTITLE sl sc
text of subtitle /E/
```

where "sl" is the line on which the subtitle is to appear and "sc" is the column position from the left text margin in which it is to start. The defaults are sl=1 and sc=1. The line(s) following the control phrase contain the text of the subtitle, as for a title. The subtitle is always printed, if one is defined, whether or not the title is, and cannot be cycled (see below) for right- and lefthand pages as the title can. If the user wants to have a cycled subtitle, a multiple-line title must be used instead of a subtitle.

A footer or running foot may appear at the bottom of each page. A footer is defined by the control phrase sequence

```
FOOTER
text of footer /E/
```

The line(s) following the control phrase line contain the footer text, terminated by the /E/ command operand. The footer is placed at the bottom of the page. The bottom is the last line as specified by the LINES control phrase; the body of the text is ended as many lines before that as are necessary. If blank lines are to be inserted between the text and the footer, they must be defined as part of the footer. By default, the text of the footer starts at the left text margin. For example, the following produces a footer aligned with the right margin (/Q/) with two blank lines (/J2/) inserted between it and the text of the document.

April 1977

```
FOOTER
/J2/ a footer /QE/
```

The following sequence of control phrases

```
WIDTH 20
TEXT 3
LINES 9
FOOTER
/J/ Mark Twain /QE/
PAGE 2 1
TITLE
Sketches /E/
```

produces pages like

```
|                      |
| Sketches          2  |
|                      |
| Soap and education   |
| are not as sudden as |
| a massacre, but they |
| are more deadly in   |
| the long run.        |
|                      |
|          Mark Twain  |
|                      |
```

FORMAT can also produce output with right- and lefthand pages. This allows, for instance, the page number always to appear on the outside, or titles to appear only on righthand pages. This is enabled by the control phrase

CYCLE

If this control phrase is given, pages are taken as alternating right- and lefthand pages, beginning with the right (even if the page number is even). On a righthand page, the page number appears in the same place as before; on a lefthand page, it appears "pc" column positions from the left text margin. Thus, if pc=0 (the default), the page number is aligned with the right margin on righthand pages, and with the left text margin on lefthand pages. The title for lefthand pages is set with the LTITLE control phrase, and that for righthand pages with the RTITLE control phrase. The format of these control phrases is identical to the format of the TITLE control phrase. Similarly, the right- and lefthand footers are set with the RFOOTER and LFOOTER control phrases, respectively. RFOOTER and LFOOTER have the same format as FOOTER. Neither right titles nor right footers are automatically right-aligned. If a TITLE or FOOTER control phrase is used, it defines a right title or footer if the current page is a righthand page, and a left title or footer, otherwise. The LEFT and RIGHT control phrases force the next

page to be a lefthand or righthand page, respectively; alternation continues in the normal way from there.

The following control phrases

```
CYCLE
WIDTH 50
RTITLE 2
@A @CONNECTICUT @YANKEE /E/
RFOOTER
@MARK @TWAIN /QE/
LFOOTER
@MARK @TWAIN /E/
LINES 9
```

would produce pages formatted as

```
|--------------------------------------------------|
|                                                1 |
| A Connecticut Yankee                             |
|                                                  |
|                                                  |
| And I promised, with my hand on my heart, that if |
| all who felt no emnity toward me would come      |
| forward and pass before me they should see that  |
| only those who remained behind would be struck   |
|                                       Mark Twain |
|--------------------------------------------------|

|--------------------------------------------------|
| 2                                                |
|                                                  |
|                                                  |
|                                                  |
| dead. The procession moved with a good deal of   |
| promptness. There were no casualties to report,  |
| for nobody had curiosity enough to remain behind |
| to see what would happen.                        |
| Mark Twain                                       |
|--------------------------------------------------|
```

Footnotes and Figures

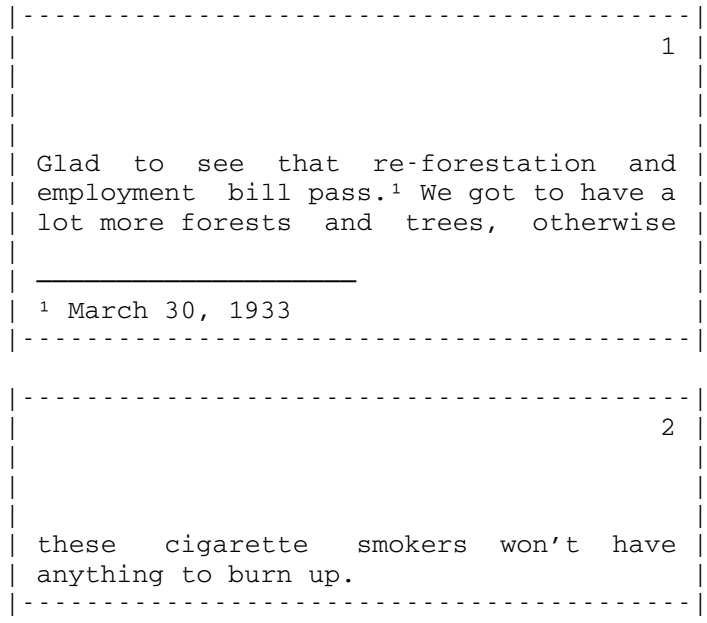
FORMAT allows a footnote to be specified anywhere within the text which is subsequently placed at the bottom of the page. A footnote is defined by the control phrase sequence

```
FOOTNOTE fs
footnote text /E/
```

where "fs" is the number of lines to insert between the last text line of the page and the footnote (if it is the first footnote on the page). The line(s) following the control phrase contain the text of the footnote, terminated by the /E/ command operand. The footnote is printed at the bottom of the current page, unless there is insufficient room, in which case it is printed as a footnote on the following page. If "fs" is omitted, one blank line is inserted at the end of the text. If "fs" is specified, a separation line is inserted before the footnote if it is the first footnote. Footnotes are always single-spaced, regardless of the spacing of the text. (See the COUNT control phrase and the counters special operator for a convenient way to number footnotes and their references.) The total number of lines of footnotes on any single page may not exceed 20. The following example uses counters to reference the footnote. The /V/ command operand is used to enter control phrase mode in order to enter the FOOTNOTE control phrase, and the GO control phrase is used to return to text mode. For example,

```
/S/@glad to see that re-forestation and employment bill pass.|F0 /V/
FOOTNOTE 1
|F1 @march 30, 1933 /E/
GO
@we got to have a lot more forests and trees, otherwise these
cigarette smokers won't have anything to burn up.
```

produces the following:



A figure is specified by the user and held by FORMAT until there is sufficient room to print it on a single page. The figure may be printed

immediately, or it may be deferred for several pages. If it is deferred, the text that appears immediately after it in the input will appear before it in the output. Typically, a figure is used for a diagram, or to provide space for later insertion of a photograph or graph. A figure is specified by the control phrase sequence

```
FIGURE fb fe
text of figure /E/
```

where "fb" and "fe" are the minimum number of lines of the page that must be above and below the figure on the page, respectively. The default values of "fb" and "fe" are chosen so that the figure may be printed anywhere within the text area of the page. "fb" and "fe" may be specified to force the figure to be printed farther up or down the page. The line(s) following the control phrase contain the text of the figure, terminated by the /E/ command operand. The following example shows a figure consisting of a caption and blank space left for insertion of a graph.

```
as shown in figure 3.  /V/
FIGURE 4
/J10/Figure 3.  Graph of numbers of dinosaurs vs.
years for the late Jurassic /E/
GO
Notice that the changes are in numbers
```

Counters (see the COUNT control phrase and the counters special operator) are often useful for numbering figures.

Text Formatting

FORMAT has several control phrases that govern the manipulation of the text. This includes paragraph spacing, word and sentence spacing, tab and indent setting, capitalization, underscoring, and several other functions.

Paragraph spacing is set by two control phrases. The default (when neither is used) is that one blank line is inserted between paragraphs, and the first line of each paragraph is indented 5 spaces. The spacing between paragraphs may be changed with the control phrase

```
SEPARATION lp
```

where "lp-1" is the number of blank lines to be inserted (thus, the default is lp=2). If lp=0, text continues on the same line. The indentation of the first line of a paragraph is set by the control phrase

```
PARAGRAPH sp
```

where "sp" is the number of spaces from the left margin to indent. The default is sp=5. The /P/ command operand starts a new paragraph. For example,

```
SEPARATE 1
PARAGRAPH 3
GO
/P/"Explain all that," said the Mock Turtle. /P/"No, No!
The adventures first," said the Gryphon in an impatient
tone: "Explanations take such a dreadful time."
```

produces

```
|    "Explain all that," said the Mock Turtle.                |
|    "No,  No!  The adventures first," said the Gryphon in an |
| impatient tone: "Explanations take such a dreadful time."   |
```

The spacing between lines is set by the control phrase

SPACING ls

where "ls" is the spacing: 1 means single-space, 2 means double-space, etc. The default is single-spacing. The spacing between words is set by the control phrase

WORD sw

where "sw" is the minimum number of blanks to be inserted between consecutive words. The default is sw=1. More than the minimum may be used if justification is enabled (the default). This control phrase is useful mainly for headings.

Spacing between sentences is set by the control phrase

SENTENCE ss

where "ss" is the number of blanks to insert (if justification is enabled, this is the minimum). The default is ss=1. A period, exclamation point, or question mark followed by either a blank or by a quotation mark and a blank indicates the end of a sentence.

Capitalization is controlled by a control phrase, a special operator, and several command operands. By default, the first character of the first word of each sentence is capitalized; this includes the first word of the document, the first word after each end-of-sentence as described above, and the first (text) word following a P, R, S, or V command operand). This feature may be disabled with the NO CAPITALIZE control phrase, and reenabled by CAPITALIZE. The /F/ command operand capitalizes the first letter of every succeeding word until disabled; the /@/ command operand causes all succeeding words to be printed in uppercase

until disabled. /O1/ may be used to prevent a letter from being capitalized. The @ special operator capitalizes the letter immediately following it. See the sections "Command Operands" and "Special Operators" for further details.

FORMAT produces justified text (aligned right and left margins) by default. The NO JUSTIFICATION control phrase is used to disable this feature. When justification is enabled, each output line is filled as far as possible with words (i.e., until the next word in the text will not fit), and then extra blanks are distributed between words to justify the line. When justification is disabled, the extra blanks are not inserted. JUSTIFICATION reenables justification. The JUST function may be used to justify a piece of text within a line (see the section "FORMAT Functions" for further details).

The /U/ command operand is used to underscore a segment of text. An occurrence of /U/ causes succeeding text to be underscored until it is disabled (another occurrence of /U/ disables underscoring). Blanks occurring in that text are by default not underscored; if the UNDERSCORE control phrase is used, blanks within an underscored text segment are underscored. The _ special operator causes the letter immediately preceding it to be underscored (see the sections "Command Operands" and "Special Operators" for further details of /U/ and _).

Tabs are used in FORMAT similar to the way they are used on a typewriter. The TABS control phrase sets the tab stops, and the /T/ command operand does the tab operation. The TABS control phrase has the format

TABS t1 t2 ...

where "ti" is the column of the i-th tab stop. The "ti" are defined with respect to the left text margin. The "ti" must be given in increasing order. Once the TABS control phrase has been given, /T/ can be used in the text to skip to the next tab stop, or /Ti/ to skip to tab stop "i" (column "ti"). Text entered into the document by a tab operation is not affected by line justification. The tab operation can also be used to insert row of dots (or other characters) between the current text position and the tab stop position. The character to be inserted is specified by the control phrase

DROP dc

where "dc" is the character to be inserted, written as a single character in enclosed primes. The default character is '.'. When a drop character is to be inserted, the /D/ or /Di/ command operand must be used instead of /T/ or /Ti/. This is most often useful for tables. For example,

```
TAB 10 40
DROP '-'
GO
Paphiopedilum /T/Price
/T/P.  'Meadow Sprite'/D/$12.00/L/
/T/P.  niveum /D2/$10.00/L/
```

produces

```
              Paphiopedilum                          Price
                       P. 'Meadow Sprite' ----------$12.00
                       P. niveum ------------------$10.00
```

Indents allow the text margin to be narrowed for part of the page. The new right and left margins to be used when indenting are defined by the control phrase

```
INDENTS (l1,r1) (l2,r2) ...
```

where each (ln,rn) is a left-right margin pair. "ln" and "rn" are the number of columns to indent (not the number of the column) from the left and right margins, respectively. If indenting is to be done from only the left or right margin, the other "rn" or "ln" should be set to zero. The /In/ and /Hn/ command operands are used to begin the actual indention in the text; the "n" specifies that the n-th indent pair is to be used as defined by the INDENTS control phrase. If a different pair is currently in use, the n-th pair replaces it. If the same pair is currently in use or if n=0, indenting is disabled. /In/ begins indenting on the current line; /Hn/ begins indenting on the next line. Both continue indenting until disabled. The following example illustrates simple use of indents:

```
INDENT (3,6) (7,6)
WIDTH 60
GO
In /U/Deerslayer/U/ Cooper violates eighteen of them.  These
eighteen require:  /LI1/1.  /I2/ That a tale shall accomplish
something and arrive somewhere.  But the /U/Deerslayer/U/
tale accomplishes nothing and arrives in the air.  /I0/
```

This produces the output

```
| In Deerslayer Cooper violates eighteen of them.  These |
| eighteen require:                                      |
|    1.  That a tale shall accomplish something and      |
|        arrive somewhere.  But the Deerslayer  tale     |
|        accomplishes nothing and arrives in the air.    |
```

FORMAT provides counters for numbering footnotes, figures, and the like. Each counter has an integer value; whenever a counter is referenced, its value is printed, and then it is incremented by one. In

addition, two of the counters have their values automatically reset to one at the beginning of every page, making them particularly useful for numbering footnotes. Some of the counters print as superscripts, while the others print as normal numbers. The 6 counters are referenced (in text mode) by |Fi, where "i" is a digit between 0 and 5, inclusive. The particulars for each counter are:

Counter	Printing	When Reset
|F0, |F1	superscript	each page
|F2, |F3	superscript	not reset
|F4, |F5	normal	not reset

The counters may be used in pairs: |F0 with |F1, |F2 with |F3, and |F4 with |F5. One member of a pair is used in the "definition" of a figure or footnote, and while other in used as the reference. This allows only one definition and one reference per value (the common case for footnotes). The footnote example given earlier illustrates the use of F0 and F1.

Any counter may be reset by the control phrase

```
COUNTERS c0,c1,c2,c3,c4,c5
```

where "ci" is the value to be assigned to "|Fi". If no value is given for a particular counter (but the comma is not omitted), that counter is not reset. For instance,

```
COUNTERS ,12,,14
```

resets |F1 to 12, |F3 to 14, and leaves the others unchanged (note only the trailing commas may be omitted). For example,

```
COUNTERS ,,,,16,16
GO
as shown in Figure |F4. /V/
FIGURE
<text of figure> /L/ Figure |F5. /E/
```

produces

> as shown in Figure 16.
> <text of figure>
> Figure 16.

Additional flexibility may be gained by using the COUNT function. See the section "FORMAT Functions" for further details.

Boxes are combinations of horizontal and vertical lines which are useful for constructing tables and charts. A box is defined by the control phrase

BOX n b1 ... b20

where "n" is the identification number of the box (1≤n≤10), and "bi" are columns (column 1 is the left margin) in which vertical lines are to be drawn (at most 20 vertical lines may be specified). The /Bn/ command operand is used to begin or end a box (box "n"). To begin a box, a horizontal line is drawn between the two outermost columns as specified on the BOX control phrase; vertical lines are then started at each column position "bi" specified. The vertical lines are continued on each succeeding output line until the box is ended. To end a box, another horizontal line is drawn between the outermost columns and the vertical lines are terminated. A box may be ended and then restarted on the same line; this will be necessary if a composite box is to be drawn which contains several horizontal lines. For example,

```
WIDTH 65
TAB 12 30
INDENT 29 5
BOX 1 11 28 62
GO
/B1LT/ TERM /T/DEFINITION /LB1B1L/
/T/DAWN /I1/The time when men of reason go to bed.  /I0LB1B1L/
/T/PROOF-READER /I1/A malefactor who atones for making your
writing nonsense by permitting the compositor to make it
unintelligible.  /I0LB1l/
```

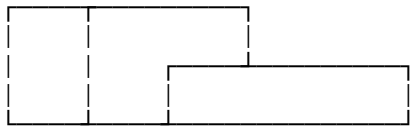
produces the output

TERM	DEFINITION
DAWN	The time when men of reason go to bed.
PROOF-READER	A malefactor who atones for making your writing nonsense by permitting the compositor to make it unintelligible.

As many as ten boxes may be defined at any one time, and any or all may be in use at any time. For example,

```
BOX 1 25 30 40
BOX 2 35 -40 50
GO
/B1J2B2J2B2B1/
```

produces



FORMAT has a facility for printing revision bars on a document. Text to be marked is preceded and followed by a /|n/ command operand, where "n" is the version number to be assigned to the text. If the

VERSION v

control phrase is used, any text which is version "v" or greater is marked with revision bars.

Listing Control

In addition to the output text, FORMAT produces auxiliary information to aid the user in modifying the document. This information includes the control phrase listing, the listing of input, the printing of input source line numbers along side the output text, and listing of error messages.

After the output text is printed, a listing of the control phrases processed is printed. For each control phrase, the source line number, the output page number (the number of the page of output text being produced when the control phrase was encountered), the actual text of the line, error indications (if any), and the interpretation of the control phrase made by FORMAT are given. If the NO CONTROL control phrase is given, no further entries are made in this listing until a CONTROL control phrase is encountered, at which time the listing resumes. If the last instance of this control phrase is NO CONTROL, the listing is not produced.

FORMAT can also produce a listing of the source lines. If produced, this listing is mixed with the control phrase listing; for each input line, the listing has a line containing the source line number, the page number (as for control phrase listings), and the text of the line. The LIST control phrase is used to enable this listing; NO LIST disables it. If the listing is disabled, only lines in which errors are detected are listed (along with the error messages).

The FULL control phrase controls the inclusion of lines generated by macro expansions in the control phrase listing. If FULL is specified, each line generated by a macro expansion is printed in the listing in the same format as a source line, except that a "+" is added in front of the text to indicate macro expansion. This listing ends when a NO FULL control phrase is encountered. This listing is produced only when source listing (LIST) is enabled.

If the NUMBER control phrase is specified, source line numbers are printed to the right of the output text. This numbering may be of help in making modifications to the source file. Numbering is enabled by the NUMBER control phrase and disabled by the NO NUMBER control phrase.

Normally, error messages (together with the erroneous input) are printed on SPRINT at the end of the document. If the ERRORS control phrase is specified, they are printed instead on SERCOM as they occur.

Input/Output Details

Several control phrases are available for specifying the details of input such as case conversion and concatenation, and details of output such as the type of printer to be used.

By default, all incoming text is translated into lowercase, so that all text appears in the output as lowercase, regardless of whether it was upper- or lowercase on input; letters may be capitalized with the /@/ command operand and @ special operator, and capitalization of the first letter in each sentence is provided. This translation may be disabled by the LOWERCASE control phrase. If the LOWERCASE control phrase is used, subsequent text lines are not translated, and uppercase letters remain as such (automatic capitalization of the first letter of each sentence is a separate issue--see the section on text formatting control phrases). The NO LOWERCASE control phrase may be used to reenable translation. For example, the following input

```
GO
I have yet to see ANY problem, however complicated.  /V/
LOWERCASE
GO
which, when you looked at it the RIGHT way, /V/
NO LOWERCASE
GO
did not become still MORE complicated.
```

produces as output

```
| I have yet to see any problem, however  complicated,  which, |
| when  you  looked  at it the RIGHT way, did not become still |
| more complicated.                                            |
```

Occasionally it may be useful to restrict the input processed to only a certain part of the input line, or to break words over input lines. A limited facility for doing this is provided by the control phrase

CARD x y

If CARD is specified, only columns "x" through "y" of each input line of text are processed; however, control phrase lines are processed in their entirety. The defaults are x=1 and y=255. If only one number is specified on the CARD control phrase, it is taken "y" and "x" is defaulted to 1. The character in column "y" of a line (if any) immediately precedes the character in column "x" of the following line--no blank is introduced between them. This feature makes CARD 1 80 especially suitable for input on cards.

The GENERATE control phrase may be used to force each input text line to become a separate output text line. Normally, the input line boundaries have no effect on output lines; if an input line does not fill an output line, part of the next input line is used. However, when this control phrase is used, each input line is treated as though it began with /L/ so that each input line begins a new output line.

The INCLUDE control phrase produces the same effect as the "$CONTINUE WITH fdname RETURN" delimiter. The format is

INCLUDE fdname

The major difference is that INCLUDE produces an entry in the control phrase listing, thus documenting the inclusion.

The TRAIN control phrase is used to specify the characteristics of the output device. The default is chosen to be compatible with a TN printer; hence, this control phrase need not be given if the TN printer is used. If the output is intended for a device with the PN character set, the TRAIN control phrase must be used. This prevents the use of lowercase letters and special characters such as box corners and superscripts. Special characters which have no similar PN character are printed as asterisks (see Appendix D, "Special Characters").

Often it is useful to use one character to represent another on input, such as when a frequently needed character is unavailable on at a terminal, or to use underscoring for overprinting, such as to produce ⊮. The TRANSLATE control phrase is used to make these character changes. Use of this control phrase is rather complex (see the section "Changing the Translate Tables").

Macros and Functions

A macro is a method of assigning a name to a segment of commonly used or difficult-to-type text, so that it will be inserted into the text, possibly with variations, whenever the name is referenced. Macros can be used to generate titles and page layout. The macro text may contain control phrases and/or input text.

Before a macro can be used, it must be defined and a name given to it. The name may be from 1 to 32 characters in length, and may contain

alphabetic characters, numerals, and periods. The first character must be an alphabetic. In macro names, the uppercase and lowercase form of a letter are considered to be identical. There are two ways to define a macro: with the SET control phrase and with the DEFINE control phrase. The SET control phrase is used to define macros with short text segments, while DEFINE is used for longer macros and those which involve both text and control phrases.

The form of the SET control phrase is

```
SET name = 'text'
```

where "name" is the macro name, and "text" is the text segment enclosed in primes. The blanks around the equal sign must be present. For example,

```
SET UM = '@university of @michigan'
```

is a simple macro definition. The text segment may contain command operands and/or other macro function calls. Since this definition must appear all on one line, the text segment cannot contain both text and control phrases.

A method of creating longer and more complex macros is provided by the control phrase

```
DEFINE name n
```

where "name" is the name of the macro. If "n" is present, the next "n" lines of input become the text segment of the macro. If "n" is omitted, the lines of the text segment follow the control phrase, each beginning with a "|" (the vertical bar does not become part of the text segment). When a macro is invoked, each line of the text segment is taken as if it were a separate line of input. Thus, the text segment can contain text and control phrases. For example, when the macro defined by

```
DEFINE RECIPE
|/L2@/ INGREDIENTS /L@V/
|BETWEEN 4
|COLUMN 2
|GO
```

is invoked, the word INGREDIENTS is printed on a separate line, and subsequent text is printed in 2-column format.

A macro is invoked in control phrase mode by using the control phrase

```
CALL name
```

where "name" is the macro to be invoked. The first line in the text segment of a macro invoked in this fashion should be a control phrase. After the macro invocation is finished, FORMAT will be in the mode in which the macro invocation finished.

A macro is invoked in text mode by specifying

"|name|" or "|name "

in a text line. Note that the name is terminated by a "|" or blank. The first form can be used to invoke the macro in the middle of a word. The text segment of a macro invoked in this way should begin in text mode. The mode after the invocation is determined by the macro. A possible invocation of the RECIPE macro is

```
/M/ Chocolate Chip Cookies /ML/
. . . Bake on greased cookie sheet at 375|31F for
10-12 minutes. |RECIPE| 1 c shortening /L/3//4 c
sugar /L/3//4 c brown sugar /L/1 tsp vanilla
/L/ 2 eggs /L/2 1//4 c flour /L/1 tsp baking
soda /L/1 tsp salt /L/12 oz chocolate chips
```

The output produced is

Chocolate Chip Cookies

. . . Bake on greased cookie sheet at 375°F for 10-12 minutes.

INGREDIENTS

1 c shortening	2 1/4 c flour
3/4 c sugar	1 tsp baking soda
3/4 c brown sugar	1 tsp salt
1 tsp vanilla	12 oz chocolate chips
2 eggs	

The following example shows a trivial macro intended to be called from text mode. The macro enters control phrase mode and issues a FIGURE control phrase. The text following the macro invocation, up to the /E/ command operand, becomes the figure. FORMAT is then in control phrase mode; thus, a GO must be used to reenter text mode.

```
DEFINE FIG
|/V/
|FIGURE 5
GO
The following figure |FIG
<text of figure> /E/
GO
shows ... /L/
```

The output from this macro invocation is

```
The following figure shows ...
<text of figure>
```

As described so far, a macro produces the same text each time it is invoked. Parameters may be used to produce variable text from a macro. A "formal parameter" is used in the definition of the text segment. In

the call to the macro, text is specified to replace the formal parameters. When a macro is invoked, wherever a formal parameter appears in the definition, the text specified in the invocation is used. A macro definition may use up to 9 formal parameters; they are referred to by the names |PAR.1|, |PAR.2|, ..., |PAR.9|. As with macro invocations, the last "|" may be omitted, if the next character is blank. In a macro invocation, the text to replace the formal parameters is specified by enclosing it in parentheses immediately after the macro name, separating the text for individual parameters by commas. The form is

```
name(par1text,par2text,...,par9text)
```

It is not necessary to specify text for all parameters (especially not for those which are not used in the macro definition). The text for a parameter may be omitted by giving only the comma and not specifying the text. Trailing commas may be omitted. In the following example, only the first and third parameters have text specified:

```
name(text,,moretext)
```

If, in the invocation, no text is specified for a parameter which is used in the definition, a null string is placed in the resulting text (thus, the text appears as if the parameter were never there). Text specified for a parameter which is not referenced is ignored. For example, the macro definition

```
SET TITLE = ')W5UL3 |PAR.1 )UL2'
```

might be invoked with

```
|TITLE(Pteranodons)
```

to produce the same result as specifying

```
)W5UL3 Pteranodons )UL2
```

in the input text.

Since a macro invocation is terminated by a blank, some special notation is needed to permit blanks to be used in text strings in the invocation. If a text string contains blanks, the entire string must be enclosed in primes. Another invocation of the title macro above might be:

```
|TITLE('The Pteranodon and Related Dinosaurs')
```

A macro invocation must be contained within a single input line. A macro definition may include invocations of other macros, but a macro may not recursively invoke itself.

The following example defines and uses a macro FIG for producing figures. FIG takes two arguments: the text of the figure and a figure

number. FIG defines the figure, putting the figure number on the last line. FIG then makes a table of contents entry using the FORMAT function CONTENTS (see the section "FORMAT Functions" for further details).

```
DEFINE FIG
|/V/
|FIGURE
||PAR.1 /L/@figure |PAR.2|./E/
|CALL CONTENTS('|PAR.1',1,'|PAR.2|.')
|GO
GO
The following figure |FIG('<text of figure>',16)
shows ... /L/
```

The output produced is:

```
The following figure shows ...
<text of figure>
Figure 16.
```

Normally, once a macro is defined, it remains defined until the same name is used to define a new macro (i.e., until it is redefined). A macro may be undefined by the control phrase

ERASE name

where "name" is the macro to be undefined. After this control phrase is given, "name" is no longer recognized as a macro name. The CLEAR control phrase may be used to undefine all macros.

A function is invoked in the same way as a macro, but, instead of there being a text segment with possible substitutions, a function is actually a program which is called. A function may produce text to be input to FORMAT. For example, there is a function named CONTENTS, which is designed to be called many times during the processing of a document to place entries into a table of contents, and then once at the end to produce the table. The function saves each of the entries, and then, on the last call, arranges the entries and generates FORMAT input to actually produce the table.

Functions are available for producing an index or table of contents, doing Roman and other special types of page numbering, producing block letters, centering, using special counters, getting the date, etc. The user can also write his own functions. (See the sections "FORMAT Functions" and "User Defined Functions" for details.)

CONTROL PHRASES

This section describes each control phrase, including the defaults and legal values for each argument.

Each control phrase must be entered on a separate line; control phrases may not be split over lines. The general format is

[NO] name [arguments]

where "name" is the name of the control phrase, and "arguments" is one or more optional arguments as appropriate for the particular control phrase. For some control phrases, the control phrase actually refers to a switch and the optional "NO" determines its setting. If "NO" is specified, the function of that control phrase is disabled until the user reenables it by repeating the control phrase without the "NO". Brackets always indicate optional arguments. Except where noted, arguments must be separated by blanks and/or commas. In addition, the user may intersperse extra words, if desired, as an aid in remembering the control phrase. However, the legal name or abbreviation of the control phrase and the NO option must always appear first. Any additional words are sometimes referred to as "noise" words. They are completely ignored, and serve only to help the user remember. For example, the following control phrases are identical:

```
PAGE 101 4
PAGE STARTING AT 101 ON LINE 4
PAGE 101,4
```

Any number of blanks may appear wherever a single blank is valid. Either "NO" or "name" must be the first word on the line, but it may be preceded by blanks. "NO" is permitted with any control phrase, and is ignored if it is inapplicable. Control phrases may be abbreviated to the first three characters, except where two control phrases have the same abbreviation, in which case more characters are required. The minimum acceptable abbreviation for each is underscored. If fewer arguments than expected are given, the ones given are assumed to be the first ones. The values taken for those not given are the defaulted unless otherwise noted. For example, the arguments for the PAGE control phrase are (in order) the first page number to use, the line on which to print it, and the number of spaces from the right margin to print it. These have defaults 1, 1, and 0. If

```
PAGE 20 2
```

is entered, 20 is the first page number, 2 is the line number, and a default value of 0 is assumed for the third argument. (The CARD control phrase is an exception to this rule.)

Initially, FORMAT is in control phrase mode. Control phrase mode is exited (text mode entered) with the GO control phrase, and entered (text mode exited) with a /V/ command operand. While in control phrase mode,

the order of control phrases is generally irrelevant; exceptions are noted in the descriptions. Once in control phrase mode, control phrases are read until either the GO control phrase or a line which is not a control phrase is encountered. After the GO control phrase, text mode is entered. In the other case, FORMAT assumes that the line is actually text, enters text mode, and takes the line as text.

Control phrases may be entered in upper- and/or lowercase.

The first line of each description below is the control phrase, preceded by NO if applicable, the arguments, and the default values. The necessary part of each name is underscored; arguments to be replaced by values are in lowercase. The second line contains the control phrase with extra words which may be useful in remembering the name. But, note, if the user introduces extra words, the first word of the control phrase must still be a legal control phrase.

BETWEEN g1 g2 g3 g4 g5 Default: gi=2
BETWEEN COLUMN GAP IS g1 g2 g3 g4 g5 PRINT POSITIONS

BETWEEN specifies the number of print positions to be inserted between columns of text (i.e., the width of the "gutters"). "gi" is the space between the i-th and (i+1)st columns. Each "gi" must be in the range of 0 to 20, inclusive. If there are "j" columns, only g1 through g(j-1) need be specified; extra values are ignored. The values given take effect with the next COLUMNS or WIDTH control phrase (thus, the BETWEEN phrase must precede COLUMNS or WIDTH).

BOX n b1 b2 ... b20
BOX n = b1 b2 ... b20

This defines box number "n", with vertical lines in columns b1, b2, ..., b20. Only those "bi" which are actually used should be specified (up to 20 columns may be specified). Columns must be specified in ascending order. "n" must be in the range of 1 to 10, inclusive. If a minus sign precedes a "bi", then no vertical line is drawn in the box; this may be used to suspend the printing of a vertical line defined for another box.

CALL name[(parlist)]

This invokes the macro "name", optionally with the parameter list "parlist". Items in the parameter list are separated by commas, and blanks are not permitted within the parameter list (except within a string enclosed in primes). FORMAT remains in control phrase mode unless the macro generates a GO control phrase.

[NO] CAPITALIZE Default: CAPITALIZE
CAPITALIZE AUTOMATICALLY

When this option is enabled, the first character of the first text word following any of the command operands C, O4, P, R, S, or V, or

following text ending in .␣, ?␣, !␣, ."␣, ?"␣, or !"␣ (␣ denotes one or more blanks) is capitalized.

CARD x y Default: x=1, y=255
CARD FIELD x THROUGH y

On text input, only columns "x" through "y" of any text line are processed; characters outside this range are ignored. Control phrase lines are still processed in their entirety. "y" must be in the range of "x" to 255, inclusive. In text mode, if y is less than 255, the characters in column "y" on one card (if any) are immediately followed by the character in column "x" of the next; no blanks are introduced between them. If only one argument is given, it is taken to be "y", and the default is taken for "x". If y=255, FORMAT returns to the normal behavior of inserting a blank at the end of each line as it is read. CARD is useful when cards are used for input, and when the input contains sequence ids.

CLEAR
CLEAR MACRO DEFINITIONS

All functions and macro definitions are deleted.

COLUMNS c Default: c=1

This control phrase sets up "c" equal-width text columns, using the value specified in the last WIDTH control phrase and with gutters at least as wide as specified on the last BETWEEN control phrase. "c" must be in the range of 1 to 6, inclusive.

COMMENT [text]

The text of this control phrase is printed in the control phrase listing. This allows optional text to be printed for documentation and reference purposes.

[NO] CONTROL Default: CONTROL
CONTROL PHRASE LISTING

This controls the listing of control phrases which appears at the end of the document. This listing includes each line read in control phrase mode and the interpretation made of it by FORMAT. CONTROL may be enabled and disabled (NO CONTROL) as often as desired within a document; only those control phrases encountered while CONTROL is enabled appear in the listing. If it is disabled at the end of the document, the listing is not printed.

COUNTERS c0,c1,c2,c3,c4,c5

This control phrase sets the counters |F0,...,|F5 to the values specified by c0,...,c5. Only the "ci" for which values are specified are reset. To omit a value, only the comma should be specified (thus leaving two consecutive commas). Commas other than

these may be replaced by blanks. Each "ci" must be in the range of 0 to 32767, inclusive. Counters are useful for keeping track of the sequence of footnotes on a page or the number of figures in a document, etc. Initially, the counters are set to the value 1; in addition, F0 and F1 are reset to 1 at the beginning of each page.

[NO] CYCLE Default: NO CYCLE
CYCLE THE PAGE NUMBER

When CYCLE is enabled, the pages are formatted as alternating righthand and lefthand pages. On a righthand page, the page number is printed relative to the right margin, and the right title and footer are used (see RTITLE and RFOOTER). On a lefthand page, page number is printed relative to the left margin, and the left title and footer are used (see LTITLE and LFOOTER).

DEFINE name [n]

This defines the macro "name". If the optional "n" is specified, the following "n" input lines are taken as the text segment of the macro. If "n" is omitted, each subsequent line beginning with a "|" is taken as part of the macro definition until a line not beginning with a "|" is encountered.

DROP dc Default: dc='.'=4B

This specifies "dc" as the drop character to be used by the /D/ command operand. "dc" may be either a single character enclosed in primes or a two-digit hexadecimal number.

ERASE name

This erases the definition of the macro or function "name". If "name" is T or D, the original definition (time or date) is again used.

[NO] ERRORS
ERRORS IN LINE Default: NO ERRORS

When ERRORS is enabled, the control phrase, source, and error listing that is normally printed on SPRINT at the end of the document is printed instead on SERCOM. Lines are printed as they are processed by FORMAT; they are not held to the end.

FIGURE fb fe Defaults: see below
FIGURE fb LINES DOWN AND fe LINES UP

This control phrase defines a figure. The line(s) following the control phrase are taken as the text of the figure, ending with the first line on which a /E/ command operand occurs that is not immediately followed by another command operand. When the figure is printed, at least "fb" lines of the page will be above it, and "fe" below it. "fb" and "fe" must be at least 0 and less than the

number of lines on the page ("l" in the LINES control phrase). The default value of "fe" is 0. The default value of "fb" is one less than "f" as given on the TEXT control phrase. If the default values are used, the figure may appear anywhere on the page that text can.

FOOTER

The line(s) following this control phrase, ending with the first /E/ command operand that is not immediately followed by another command operand, become the text of the footer. The number of footer lines on a page may not exceed 5. If CYCLE is not enabled, the footer is printed at the bottom of each page; if it is enabled, the footer is printed only at the bottom of pages of the type (right or left) of the current page.

FOOTNOTE fs Default: fs=0
FOOTNOTE LEAVING fs LINES

FOOTNOTE specifies that the following lines, ending with the first /E/ command operand encountered, contain a footnote to be printed on the current page. The footnote is single-spaced. At most, 20 lines on a given page may be used for footnotes. If "fs" is specified, "fs" blank lines are inserted and a separation line is printed before the footnote, if it is the first footnote on the page. "fs" must be less than 10.

[NO] FULL Default: NO FULL
FULL SOURCE LISTING

If FULL is enabled, lines generated by macro invocations are printed in the control phrase and source listing.

[NO] GENERATE Default: NO GENERATE
GENERATE A /L/ FOR EACH LINE

When this is enabled, each line is treated as though it began with a /L/ command operand.

GO

This causes FORMAT to leave control phrase mode and enter text mode.

[NO] HYPHENATION Default: HYPHENATION

If this is enabled, any word containing a hyphen (actually a minus sign, "-") may be split over two lines at the hyphen.

INCLUDE fdname

Input is taken from "fdname", just as if "$CONTINUE WITH fdname RETURN" were used instead of the control phrase.

INDENT l1 r1 l2 r2 ... l10 r10
INDENT COLUMNS ARE (l1,r1) ... (l10,r10)

This establishes indent pairs (ln,rn) for use by /Hn/ and /In/. At most, 10 pairs may be specified; those pairs not specified are undefined. In each pair, "ln" is the number of columns to indent from the left text margin, and "rn" is the number to indent from the right margin. If it is desired to indent from only the left or right margin, the other "rn" or "ln" should be set to zero. Note that the parentheses and commas act as if they were blanks.

[NO] JUSTIFICATION DEFAULT: JUSTIFICATION
JUSTIFICATION OF TEXT

When this is enabled, normal text is justified (aligned right and left margins) by FORMAT. Text is justified except during "as-is" mode (/A/) or when a line is "prematurely" terminated by a command operand other than /G/ (e.g., /L/ or /P/). Text is justified by inserting extra blanks as evenly as possible between words (the extra blanks are inserted alternately towards the right and left sides of successive lines).

[NO] KEYPUNCH Default: NO KEYPUNCH
KEYPUNCH PROVIDES LOWERCASE

This is a synonym for LOWERCASE. See the description of the LOWERCASE control phrase.

[NO] LEFT Default: NO LEFT
LEFT HAND PAGE

If LEFT is given, the next page is formatted as a lefthand page.

LFOOTER

The line(s) following this control phrase, ending with the first /E/ which is not immediately followed by another command operand, become the text of the footer to be inserted at the bottom of every lefthand page. All restrictions that apply to the FOOTER control phrase also apply to LFOOTER.

LINES l Default: l=60
LINES PER PAGE ARE l

The number of lines per page, including text, titles, and footers, is set to "l". "l" must be in the range of 5 to 1000, inclusive.

[NO] LIST Default: (batch) LIST; (terminal) NO LIST
LIST SOURCE LINES

When this is enabled, the control phrase and source line listing normally produced at the end of the document includes a line for each line of the source (including text lines). This line gives

the current page number, file line number, the line itself, and any error comments. When this is disabled, this information is printed only for lines in which errors are detected.

LOAD name fdname entry Default: fdname=ETC:FMTLIB, entry=name

The function "name" with entry point "entry" is loaded from the file or device "fdname". This control phrase is not necessary for library functions; they are loaded on first reference. "entry" must be a defined entry point for the function; "name" may be an alias by which the function is known in FORMAT or may be an entry point name. Any previous use of "name" is deleted. A function may be (simultaneously) loaded more than once by giving different "name"s for the same "entry". If "entry" is already loaded, "*" may be given for "fdname"; the effect is the same as if the file name had been specified.

[NO] LOWERCASE Default: NO LOWERCASE
LOWERCASE INPUT

If this is not enabled, all incoming text is translated into lowercase (text is selectively translated back into uppercase by /@/, @, etc.). If it is enabled, no translation is done; text remains in the case entered, although selective capitalization may also be done via command operands and special operators. Use of this control phrase always nullifies the effects of any previous use of the TRANSLATE INPUT control phrase.

| LTITLE tl tc Default: tl=1, tc=1
LTITLE STARTS ON LINE tl IN PRINT POSITION tc

The line(s) following this control phrase, ending with the first /E/ not immediately followed by another command operand, become the text of the title to be printed on lefthand pages. "tl" and "tc" are as defined for the TITLE control phrase. All restrictions that apply to the TITLE control phrase also apply to LTITLE.

MTS

This causes a return to MTS via the system subroutine MTS. FORMAT may be restarted (if it has not been unloaded) by a $RESTART command.

[NO] NUMBER Default: NO NUMBER
NUMBER THE OUTPUT WITH SOURCE LINE NUMBERS

When NUM is enabled, the current source (file) line number is placed in the margin to the right of each line in the output text. This facilitates making changes in the source file.

[NO] PAGE pn pl pc Default: pn=1, pl=1, pc=0
PAGE NUMBER pn ON LINE pl SHIFTED pc COLUMNS

The next page number to be used is "pn", and page numbering continues from there incrementing each page by 1. Page numbers are placed in line "pl" of each page, displaced "pc" columns from the right margin (left margin on a lefthand page). If "pn" is omitted (none of "pn", "pl", or "pc" specified) or zero, then page numbering stops if it was being done, or is restarted (at the appropriate number) if numbering was previously stopped. NO PAGE stops page numbering. Pages continue to be counted when not being numbered, so that restarting numbering results in the proper page number. A document is started with pn=1 and default "pl" and "pc", unless otherwise instructed. Any values of "pl" and "pc" are valid, but the page number is printed only if it is specified within the bounds of the page. The page number overprints any other text which occurs in the same position. If "pn" is less than 0, the current page number is not changed. This may be used to alter the position of the page number without affecting the numbering.

PARAGRAPH sp Default: sp=5
PARAGRAPH INDENT sp PRINT POSITIONS

The beginning of each paragraph (as determined by the /P/ command operand) is indented "sp" spaces.

[NO] REPEAT Default: REPEAT
REPEAT TITLE ON EVERY PAGE.

When REPEAT is enabled, the title is repeated on each page of the document. When disabled by NO REPEAT, the title is printed only once. Printing of the title may be resumed by respecifying the REPEAT control phrase.

RFOOTER

The line(s) following this control phrase, ending with the first /E/ which is not immediately followed by another command operand, become the text of the footer to be inserted at the bottom of every righthand page. All restrictions that apply to the FOOTER control phrase also apply to RFOOTER.

[NO] RIGHT Default: RIGHT
RIGHT HAND PAGE

If RIGHT is given, the next page is formatted as a righthand page-

RTITLE tl tc Defaults: t1=1, tc=1
RTITLE STARTS ON LINE tl IN PRINT POSITION tc

The line(s) following this control phrase, ending with the first /E/ not immediately followed by another command operand, become the text of the title to be printed on righthand pages. "tl" and "tc" are as defined for the TITLE control phrase. All restrictions that apply to the TITLE control phrase also apply to RTITLE.

[NO] SAVE Default: NO SAVE
SAVE MACRO DEFINITIONS BETWEEN JOBS

If SAVE is given, all macro definitions are saved between "jobs" (separate documents, separated by the /E/ command operand, may be processed in the same run of FORMAT); thus, it is not necessary to redefine them.

SENTENCES ss Default: ss=1
SENTENCES SEPARATED BY AT LEAST ss BLANKS

Sentences in the text are separated by at least "ss" blanks (or exactly "ss", if JUSTIFICATION is disabled).

SEPARATION lp Default: lp=2
SEPARATION BETWEEN PARAGRAPHS IS lp LINES

"lp-1" blank lines are inserted between paragraphs (as determined by the /P/ command operand).

SET name = 'text'

This defines the macro called "name" with text segment "text". Note that the "=", and the blanks surrounding it, must appear as well as the primes around the text segment. Noise words are not permitted on this control phrase.

SPACING ls Default: ls=1
SPACING OF TEXT IS ls LINES

"ls" defines the spacing for text. ls=1 means single-spacing, ls=2 means double-spacing, etc.

SUBTITLE sl sc Default: sl=1, sc=1
SUBTITLE STARTS ON LINE sl IN PRINT POSITION sc

The line(s) following this control phrase, ending with the first /E/ command operand, become the text of the subtitle to be printed on each page. The subtitle becomes undefined, if the text consists exactly of /E/. CYCLE does not affect the printing of the SUBTITLE. If line "sl" contains the title, the subtitle is printed on the following line.

[NO] SUPER Default: SUPER
SUPER FORMAT

When SUPER is enabled, command operands may be delimited by slashes [/operands/] or by a blank followed by a right parenthesis ")" and a blank [)operands]. Command operands delimited by slashes may appear in the middle of text words. When SUPER in enabled, two slashes (//) should be used to insert a single slash into the text; for example, a//b on input produces a/b on output. When SUP is disabled, only the)operands form is recognized, and slashes which appear in the input will be left in the text.

TABS t1 t2 ... t20

This control phrase sets the tab positions to be used with the /T/ and /D/ command operands. At most, 20 tab positions may be specified, separated by blanks. Each "ti" set must be in the range of 1 to 132. The "ti"s must appear in ascending order.

TEXT f p Default: f=5, p=5
TEXT STARTS ON LINE f IN PRINT POSITION p

The first text line of the page is printed at line "f" (the first line of the page is line 1). The title and page number, if any, may appear above the text. "p" is the left text margin, that is, the left text margin is at column "p", where the leftmost column of the page is column 1. All other column position values are determined relative to the left text margin "p" being column 1.

TITLE t1 tc Default: t1=1, tc=1
TITLE STARTS ON LINE t1 IN PRINT POSITION tc

The line(s) following this control phrase, ending with the first /E/ not immediately followed by a command operand, become the text of the title. The title can occupy at most 20 lines. The title begins on line "t1" at print position "tc" (with respect to the left text margin). The title becomes undefined, if the text consists of exactly /E/. The title is printed on every page unless NO REPEAT is used.

[NO] TRAIN Default: NO TRAIN
TRAIN ON PRINTER HAS ONLY UPPERCASE

If this control phrase is given, the output is assumed to be destined for a PN printer, instead of the default TN printer. Characters which are not available on the PN printer are replaced by similar characters or asterisks. The list of characters affected and the characters printed for them appears in Appendix D, "Special Characters."

TRANSLATE table loc c

This control phrase changes entries in three of the translate tables used by FORMAT. "table" specifies the table to be changed, "loc" is the location in the table to be altered (as a displacement from the beginning of the table), and "c" is the new value to be put at that location. "table" must be one of: INPUT, OUTPUT, UNDERSCORE; "loc" and "c" may be either a single character enclosed in primes, or a two-digit hexadecimal number. Noise words are not permitted in this control phrase. See "Changing the Translate Tables" for further details.

[NO] UNDERSCORE Default: NO UNDERSCORE
UNDERSCORE BLANKS

If UNDERSCORE is enabled, blanks which occur while the /U/ command operand is in effect are underscored. Otherwise, such blanks are not underscored. This does not affect the underscoring of nontrivial blanks.

VERSION v Default: v=0
VERSION v OF DOCUMENT

"v" becomes the current version number to be used by the "|n" command operand. "v" must be in the range of 0 to 32767, inclusive.

| WIDTH w1 w2 w3 w4 w5 w6 Default: w1=64, w2=w3=w4=w5=w6=0
WIDTH OF COLUMNS IS w1 w2 w3 w4 w5 w6 PRINT POSITIONS

WIDTH sets the number of characters in each column of text. One value should be specified for each column desired; a column is set up for each value actually given. "wi" is the width in characters of the i-th column. Gutters, as specified on BETWEEN, are inserted between the columns. Each "wi" must be greater than or equal to 10, and the total width (columns and gutters) must not exceed 255. Note the default is one column, 64 characters wide.

| WORD sw Default: sw=1
WORD SPACING IS sw BLANKS

At least "sw" blanks are inserted between words (exactly "sw", if justification is disabled).

COMMAND OPERANDS

Command operands are instructions given to FORMAT in text mode which have a local effect on the text, such as starting a new paragraph, tabbing, or skipping a line. Unlike control phrases, which for the most part give FORMAT instructions on what is to be done globally to the document, command operands take effect locally, and either have a single isolated effect, or one which continues only until terminated by another command operand.

A command operand is a single character (usually an upper- or lowercase letter), possibly followed by a number (no intervening blanks). Command operands are specified in the form of command words. A command word consists of one or more command operands concatenated (no blanks) and delimited in one of the following ways:

(1) preceded and followed by "/", e.g., /L/;
(2) preceded by ")" (blank not required at the beginning of a line), and followed by a blank or end-of-line.

The first form may be used only if SUPER is not disabled (SUPER is disabled when a NO SUPER control phrase has been specified). The second form requires a blank (or line boundary) on each side, and thus must appear only between text words; the first form requires no blanks, and may appear in the middle of text words. For example:

```
un )@U der )@U score
```

produces

un DER score

while

```
un/@U/der/@U/score
```

produces

unDERscore

/U/ is the command operand meaning "underscore text from here until another /U/ occurs"; and /@/ means "capitalize until the next /@/ occurs." When SUPER is enabled, a pair of slashes // must be used to put a literal slash / into the text.

Since command operands have local effects, the order of operands within a command word is often important. In the example above, the order of "@" and "U" is irrelevant (either /@U/ or /U@/ would do), since both start and stop printing text characters in a particular way. However, for operands which indent or skip lines, order is important. For example, /LP/ is not the same as /PL/ (P is "start paragraph"; L is "skip to the next line"). /LP/ means to skip to the next line, and then

start a new paragraph (which generally involves skipping another line and indenting). /PL/ means start a new paragraph, then skip to the next line. In the first case, text following /LP/ begins with the usual paragraph indentation, while in the second, the following text is started at the left margin.

Several command operands enable and disable certain operations. For example, in

/@/a title/@/

the first occurrence of the /@/ command operand begins capitalizing each word, and the second ends it. These command operands are:

F H I M U X ! | @ ¢

There are two other methods to disable one of these; unlike the method noted above, these do not disable an operation which is already disabled. One method is the command operand /R/, which disables all of them. The other method is placing a minus sign (-) in front of one of the above operands; this disables only that particular operand. For example, /-@/ disables capitalization.

The following is a list of the command operands:

/A/

Begin "as-is" mode. All input after /A/ on the current input line is ignored, and, beginning with the next input line, text is read and printed, line for line, without change in spacing or use of upper- or lowercase. Lines longer than the current line width are truncated (no error message is given). Command operands other than @, ¢, and U are ignored (and will appear in the text). The @, ¢, and U command operands and the special operators are processed as normal. Except for this, text appears on output exactly as it appears on input. "As-is" mode is terminated by a line beginning with ") ", or with "/" when the SUPER control phrase is in effect. For example,

```
/A/
    * /@/binomial coefficient c(n,m)/@/
    C      M = LT(N - M,M) N - M
           C = EQ(M,0) 1              :S(RETURN)
           C = N * C(N - 1,M - 1) / M :(RETURN)
/
```

produces

```
    * BINOMIAL COEFFICIENT C(N,M)
    C      M = LT(N - M,M) N - M
           C = EQ(M,0) 1              :S(RETURN)
           C = N * C(N - 1,M - 1) / M :(RETURN)
```

/B/

Backspace. This backspaces over the previous character in the current line, thus deleting it. This is most useful for deleting an unwanted blank when using the ")" form of an operand. (This sort of problem can also be eliminated by using the "/" form.) For example,

```
)U abc )UB def
```

produces

abcdef

/Bn/

Begin or end box "n" (as defined by the BOX control phrase). If box "n" has not previously been started, a beginning horizontal line is drawn between the outermost column positions and vertical lines are started in each specified column position. If box "n" has been previously started, an ending horizontal line is drawn between the outermost columns and the vertical lines are terminated. This command operand is not valid if box "n" has not been defined by the BOX control phrase.

/C/, /Cn/

Skip to the top of next column or the n-th column. If the /C/ form is used, text is started at the top of the next column; if the current column is the last one on the page, the first column is started at the beginning of the next page. If the /Cn/ form is used and "n" is greater than the number of the current column, text is started at the top of column "n" (provided one has been defined by the COLUMNS or WIDTH control phrase). If "n" is less than the number of the current column, succeeding text starts in column "n", beginning one line below the lowest line currently on the page. A new "top" is set at this line (a new top is also set at the beginning of each page, and whenever a WIDTH or COLUMNS control phrase is used). For example,

```
BETWEEN 3
WIDTH 60
COLUMN 3
GO
First column/C/Second column/C/Third column/C1/First, line 2
/C3/Third, line 2/C2/Second, line 3 /S/
```

produces

```
First column        Second column      Third column
First, line 2                          Third, line 2
                    Second, line 3
```

/D/, /Dn/

Tab, leaving the character set by the DROP control phrase. A skip is made to the next (/D/) or to the n-th (/Dn/) tab stop, inserting the drop character (the default is a period) in each position between the current column position and the tab stop.

/E/

End the text hold (title, footnote, figure, or footer) or the job. If encountered while processing a text hold, the text hold is terminated. If encountered in normal text, it marks the end of the document. In this case, the document is ended, the control phrase listing is printed, and reinitialization for processing another document is done. An end-of-file also marks the end of a document, but execution is terminated after the control phrase listing is printed. /E/ must be the last command operand in a command word.

/F/

Enable or disable capitalization. If capitalization is currently disabled (the initial state), /F/ enables it. When enabled, the first letter of each word is capitalized. If capitalization is currently enabled, another /F/ disables it. Capitalization is also disabled by the command operands S, R, P, and V. For example,

```
/F/the highwayman/F/
```

produces

```
The Highwayman
```

/G/

Justify the text on the current line and start a new line. This is useful in titles and footers to force part of the line to each margin. For example,

```
left-side right-side/G/ new line
```

produces

```
|                                |
| left-side           right-side |
| new line                       |
```

/H/, /Hn/

Change column margins beginning with the next line. The margins are set by the INDENTS control phrase; if INDENTS has not been given, or if "n" is greater than the number of indents set, use of this operand is an error. If /Hn/ is used, and the n-th set of indents is not the set currently in use, then use of the n-th pair of indents begins with the next line (the current line continues normally). If a set other than the n-th set is in use, the n-th set replaces it. If the n-th pair is in use, and /Hn/ or /H0/ or /H/ is used, indenting stops; the S and R command operands also stop indents. If indents are not currently being used, /H/ or /H0/ are equivalent to /H1/. If a pair of indents is in use by /I/ or /In/, using /H/ or /Hn/ adds the indent amounts together. For example,

```
WIDTH 60
LOWERCASE
INDENTS 5 5 10 10
GO
I love work; it fascinates me.  I can sit and look at it for
hours.  I love to /H/keep it by me; the idea of getting rid
of it nearly breaks my heart.  You cannot give me too much work;
/H2/to accumulate work has almost become a passion with me;
my study is so full of it now that there /H/is hardly an inch of
room for any more.  I shall have to throw out a wing /G/
```

produces

```
| I love work; it fascinates me. I can sit and look at it  for |
| hours.  I  love to keep it by me; the idea of getting rid of |
|      it nearly breaks my heart. You cannot give me  too       |
|      much  work; to accumulate work has almost become a      |
|           passion with me; my study is so full  of           |
|           it  now  that there is hardly an inch of           |
| room for any  more.  I  shall  have  to  throw  out  a  wing |
```

/I/, /In/

Change column margins immediately. If the left half of the indent pair is set at or to the right of the current column position, the indenting begins on the same line. If it is set to the left, text continues on the next line with the new indents. Except for when indenting begins, /I/ behaves exactly the same as /H/. For example,

```
WIDTH 60
LOWERCASE
INDENTS 5 5 10 10
GO
soon.  And I am careful of my work, too.  Why, some of the
/I1/work that I have by me now has been in my /H1/
possession for years and years, and there isn't a
finger-mark on it.  I take a great pride in my work; I take
it down now and then and /I0/dust it.  No man keeps his
work in a better state of preservation than I do./H0/ But,
though I crave for work, I still like to be fair.  I do not
ask for more than my proper share.  But I get it without
asking for it--at least, so it appears to me--and
this worries me.
```

produces

```
     soon. And I am careful of my work, too. Why, some of the
|         work that I have by me now has been in my
|              possession for years and years, and
|              there isn't a finger-mark on it. I take
|              a great pride in my work; I take it down
|              now and then and dust it. No man keeps his
|         work in a better state of preservation than I do.
     But, though I crave for work, I still like to be fair. I do
     not ask for more than my proper share. But I get it without
     asking for it--at least, so it appears to me--and this
     worries me.
```

/J/, /Jn/

Skip to a new line. If /Jn/ is specified, "n" lines are skipped (leaving n-1 blank lines). /J/ begins a new line, even if text processing is currently at the top of a page. If SPACING m has been given, /J/ causes m-1 blank lines to be inserted. For example,

```
"Remember the Golden Rule!"  /J/"What's that?"  /J/ "Whoever has
the gold, makes the rules!"
```

produces

```
"Remember the Golden Rule!"
"What's that?"
"Whoever has the gold, makes the rules!"
```

/L/, /Ln/

Skip to a new line, unless currently at the top of a page. Except for this restriction, /L/ behaves the same as /J/.

/M/

Enable or disable centering of text. If centering is not currently enabled, it is enabled. When centering is enabled, each line of text is centered within the current text margins (or indent margins, if indents are in use). If centering is currently enabled, this operand disables it. The P, R, and S command operands also disable centering. For example,

```
/M/IT IS IMPOSSIBLE TO MAKE ANYTHING
FOOLPROOF/L/BECAUSE FOOLS ARE SO INGENIOUS/ML/
```

produces

```
it is impossible to make anything foolproof
      because fools are so ingenious
```

/On/

The function performed by this command operand is defined by "n". The available functions are:

```
/O1/ - reverse capitalize-next-letter switch
/O4/ - enable capitalize-next-letter switch
/O5/ - disable capitalize-next-letter switch
/O6/ - disable fixed-column switch
```

The capitalize-next-letter switch is set by operands P, S, C, R, and end-of-sentence (.,? etc.). If the NO CAPITALIZE control phrase has been given, changing this switch has no effect.

The fixed-column switch is enabled by /Cn/, and is disabled by /C/ or the COLUMNS or WIDTH control phrase. When it is enabled, text continues in the specified column even over a page boundary. /O6/ disables this switch so that when text reaches the bottom of column "n" (if it does), it continues into the next column.

/P/

Begin new paragraph. The number of lines and spaces specified by the SEPARATION and PARAGRAPH control phrases are skipped. The functions controlled by the operands F, M, U, X, @, and ¢ are disabled. If CAPITALIZE is enabled, the first character of the next word is capitalized.

/Q/

Align right. Shift all text on the current line to the right margin, then skip to a new line. The most common use of this operand is to produce right titles and right footers. For example,

```
RFOOTER
a right footer/QE/
```

produces a right-page footer with "a right footer" appearing on the righthand edge (without the /Q/, it would appear on the lefthand edge).

/R/

Disable the functions performed by the command operands F, H, I, M, U, X, @, and ¢. If text processing is not currently at the beginning of a line, a new line is started. If CAPITALIZE is in effect, the first character of the next text word is capitalized.

/S/

Begin new page, if not already at the beginning of a page. The functions performed by the command operands F, H, I, M, U, X, @, and ¢ are disabled. If CAPITALIZE is in effect, the first character of the next text word is capitalized. /SJS/ may be used to produce a blank page. /Wn/, where "n" is greater than the number of lines remaining on the page, may be used to skip without disabling any of the above functions. /S/ is not valid in titles, footers, footnotes, and figures.

/T/, /Tn/

Tab to the next or n-th tab stop. The tab stops are set by the TABS control phrase; this operand is not valid unless tabs have been set. If the tab is to the left of the current position, a new line is started before going to the tab stop. If the tab is to the right of the current margin, the effect is the same as a /L/ operand. If there is no next or n-th tab, use of this operand is an error. For example,

```
TABS 10 20 30
GO
THIS IS A THREE-BY-THREE MATRIX.  /T1/11/T/12/T/13/T1/21/T/22
/T/23/T1/31/T/32/T/33/L/
```

produces

```
          This is a three-by-three matrix.
                   11        12        13
                   21        22        23
                   31        32        33
```

/U/

Enable or disable underscoring. If underscoring is currently disabled, it is enabled. While underscoring is enabled, all text is underscored (blanks are underscored only if the UNDERSCORE control phrase is in effect). If underscoring is currently enabled, this operand disables it. The underscore special operator (_) may be used to underscore a single character.

/V/

Enter control phrase mode. Any text remaining on the current input line is ignored. /@/, /F/, and /U/ are disabled. If CAPITALIZE is in effect, the first character of the next text word is capitalized.

/W/, /Wn/

If fewer than "n" (output) lines remain for text on the current page, skip to the next page. /W/ is equivalent to /W1/. /W/ and /Wn/ may not be used in figures, titles, footers, or footnotes.

/X/

Enable or disable suppression of blanks. If not currently enabled, suppression is enabled. When suppression is enabled, all blanks are removed from the line, including blanks introduced by the ")" form of command operands (nontrivial blanks are not blanks and are not removed). If suppression is currently enabled, it is disabled. Suppression of blanks is disabled also by the P, R, S, and V command operands. For example,

```
a line /X/with a very long word/X/ in it.
```

produces

```
a line withaverylongword in it.
```

/@/, /¢/

Enable or disable uppercase printing. If not currently enabled, uppercase printing is enabled. When enabled, all letters are printed in uppercase. If uppercase printing is currently enabled, it is disabled. Uppercase printing is disabled also by the P, R, S, and V command operands. /@/ and /¢/ are interchangeable. For example,

```
this has nothing to do with /@/capitalize/@/.
```

produces

```
this has nothing to do with CAPITALIZE.
```

/|n/, /!n/

Enable or disable printing of revision bars. If currently enabled, printing of revision bars is disabled. If currently disabled, it is enabled; revision bars are printed if "n" is greater than or equal to the value specified by VERSION. The revision bars are not printed if the text starts in print positions 1 or 2. This is also disabled by /S/. /|n/ and /!n/ are interchangeable.

SPECIAL OPERATORS

Special operators are used by FORMAT for immediate text editing effects. There are special operators to capitalize and underscore individual letters. There is a special operator to specify characters which are not available on some terminals. Another is used to repeat a character a specified number of times. Special operators exist to print the time and date, and the value of a counter. Also, there is a hyphenation operator. Special operators appear in the text itself; they are not delimited like command operands.

Capitalization: @, ¢

A letter immediately following @ is capitalized. Any of the characters 0,1,2,3,4,5,6,7,8,9,-,+,(,) immediately following a @ are printed as a superscript. If any other character follows @, the @ is taken as a normal text character. @ and ¢ are interchangeable.

Examples:

Input	Output
@a @title	A Title
ab@cde	abCde
A@(@2@+@3@)*B	$A^{(2+3)}$*B
A@*D	A@*B

Underscoring: _

The character immediately preceding an _ is underscored. There is one exception: if the character is a blank it is not underscored unless the UNDERSCORE control phrase is in effect. An underscore may be inserted into the output text by underscoring the nontrivial blank, i.e., "¬_".

Examples:

Input	Output
U_N_D_ERSCORE	UNDERSCORE
INPUT¬_LOOP	INPUT_LOOP
@a_bc	Abc

Nontrivial Blank: ¬

The nontrivial blank is treated as a normal character (e.g., just as if it were a letter); but it prints in the output text as a blank. Note that this differs from a blank in that a blank separates words; wherever a blank appears, extra blanks may be inserted for justification. A period or any of the other end-of-sentence strings is not recognized as an end-of-sentence, if it is immediately followed by a "¬"; a ")" is not recognized as a command operand delimiter, if immediately preceded by a "¬". For example,

```
WIDTH 40
GO
left¬edge right¬edge/G/
```

produces

```
|                                          |
| left edge                  right edge    |
|                                          |
```

As another example,

```
The use of the nontrivial blank in "WIDTH¬40"
insures that/L/ The use of the nontrivial blank in
"WIDTH 40" insures that/L/
```

produces

```
| The use of the nontrivial blank in |
| "WIDTH 40" insures that            |
| The use of the nontrivial blank in "WIDTH |
| 40" insures that                   |
```

Special Characters: |nn, !nn

Characters which are not found on some terminals, as well as those which are special operators, may be inserted into the output by using the "|" special operator. Each "special character" is assigned a 2-digit number. The list of assignments is given in Appendix D, "Special Characters." Using the special operator "|nn" or "!nn" inserts the character whose number is "nn" into the output text. Note both digits must be given.

Examples:	Input	Output
	abc|46def	abc@def
	A|18C+D	A≤C+D
	|16NO|17 UND	[NO] UND

| Repetition: x|@nn, x!@nn

| The character "x" immediately preceding "|@nn" or "!@nn" is printed "nn" times. Both digits must be included. The |, @, or _ special operators may be used to construct the character to be repeated. The FOR function may be used to repeat a string (see the section "FORMAT Functions").

Examples:	Input	Output
	*|@03	***
|	@a!@04	AAAA
	|28|@10	¤¤¤¤¤¤¤¤¤¤
	a_|@03	aaa

Counters: |F0, |F1, |F2, |F3, |F4, |F5

The special operators |F0,...,|F5 are used to refer to the counters as set by the COUNTERS control phrase. The counters are useful for numbering footnotes and figures. Each of the counters has a value. Whenever a counter is referenced, its value is printed, and then the value is incremented by 1. All counters are initialized to 1; counters 0 and 1 are reset to 1 at the beginning of each page. The values of counters 0, 1, 2, and 3 are always printed as superscripts, while 4 and 5 are printed as normal numbers.

Example:

Input	Output
|F0 and |F0 and |F4	^{1} and ^{2} and 1

Time and Date: |D, |T

"|D" prints the current date in the form MM-DD-YY. "|T" prints the current time in the form HH:MM:SS, using a 24-hour clock. If a function or macro is defined with "T" or "D" as a name, the "|T" or "|D" special operator is unavailable until the macro or function is ERASEd or CLEARed. Other forms of the date and time may be obtained with the DATE function.

Example:

Input	Output
version of |D |T	version of 01-13-77 16:23:41

Hyphenation: |-

If "|-" appears in a word, the word may be hyphenated at that point. The "|-" will always be removed. If the word is hyphenated, a hyphen will be inserted at the appropriate point. Words will not be hyphenated if the HYPHENATION control phrase is not in effect. For example,

```
"How often have I said to you that when you have
elimi|-na|-ted the impossible, whatever remains,
/U/however improbable/U/, must be the truth."/L/
```

produces

"How often have I said to you that when you have elimina-ted the impossible, whatever remains, however improbable, must be the truth."

PRODUCING A TABLE OF CONTENTS

A table of contents may be easily generated for a document by using the CONTENTS function. The CONTENTS function remembers the entry and the current page number and upon request prints them out in the proper format at the end of the document. For example, the following lines generate a simple table of contents.

```
INDENTS 5 0
GO
/S@/ introduction /@IP/
|CONTENTS('INTRODUCTION')
...
/S@/ CHAPTER I /IP/
|CONTENTS('CHAPTER I')
...
/S@/ CHAPTER II /IP/
|CONTENTS('CHAPTER II')
...
/S@/ TABLE OF CONTENTS /L2@/
|CONTENTS(INITIALIZE)
|CONTENTS(GENERATE)
```

This would produce the following table of contents:

```
TABLE OF CONTENTS

Introduction ..............................................1
Chapter I .................................................2
Chapter II ...............................................10
```

Using Macros to Make Entries

In most documents, it is useful to define one or more macros to produce the major or minor headings required. Since in most cases the table of contents entry is the same as the heading to be printed, the macro may be defined in such a way that it invokes the CONTENTS function and automatically makes the table of contents entry. The previous example in that case would become:

```
INDENTS 5 0
SET HEADING = ' /s@/ |par.1 /IP/|CONTENTS(''|PAR.1'') '
GO
|HEADING('INTRODUCTION')
...
|HEADING('CHAPTER I')
...
|HEADING('CHAPTER II')
...
|CONTENTS(OFF)
|HEADING('TABLE OF CONTENTS')
|CONTENTS(INITIALIZE)
|CONTENTS(GENERATE)
```

The output would be exactly the same, but the input is now much simpler, and easier to modify. The CONTENTS OFF command disables the CONTENTS function and is required to ensure that the heading "TABLE OF CONTENTS" was not actually placed into the table of contents by the last use of the HEADING macro. The CONTENTS ON command (not needed here) can be used to enable CONTENTS entry again if more entries are to be made.

One problem with the normal table of contents format is that the page numbers are not right-justified. This may be arranged by using the CONTENTS PAD command which specifies how many characters are to be inserted before the page number for justification. The input then becomes:

```
INDENTS 5 0
SET HEADING = ' /s@/|PAR.1 /IP/|CONTENTS(''|PAR.1'') '
GO
|CONTENTS(PAD,3)
|HEADING('INTRODUCTION')
...
```

The table generated then appears as

```
TABLE OF CONTENTS


Introduction ..........................................  1
Chapter I .............................................  2
Chapter II ............................................ 10
```

In documents that have two or more levels of headings, an optional second argument to the CONTENTS function may be used to specify the level of the heading. When this argument is omitted (as it has been up to now), it is assumed to be 1. In the following example, the second-level headings are produced by the SUBHEAD macro.

```
INDENTS 5 0
DEFINE SUBHEAD 1
/W10L2RF/|PAR.1 /IP/|CONTENTS('|PAR.1',2)
SET HEADING = ' /S@/ |PAR.1/IP/|CONTENTS(''|PAR.1'') '
GO
|CONTENTS(PAD,3)
|HEADING('INTRODUCTION')
...
|HEADING('CHAPTER I')
|SUBHEAD('INTRODUCTION')
...
|SUBHEAD('METHODS')
...
|SUBHEAD('RESULTS')
...
|HEADING('CHAPTER II')
|SUBHEAD('INTRODUCTION')
...
|SUBHEAD('METHODS')
...
|CONTENTS(OFF)
|HEADING('TABLE OF CONTENTS')
|CONTENTS(INITIALIZE)
|CONTENTS(GENERATE)
```

This would produce the following table of contents:

```
TABLE OF CONTENTS


Introduction ..........................................  1
Chapter I .............................................  2
  Introduction ........................................  2
  Methods .............................................  3
  Results .............................................  6
Chapter II ............................................ 10
  Introduction ........................................ 10
  Methods ............................................. 12
```

Generating Tables of Figures, etc.

Often there is a need for more than one table of headings and associated page numbers in a document. Usually this is for tables of figures or examples, but there are other uses as well. Since it is quite possible to want to generate a table of contents as well, the CONTENTS function is designed to allow a number of tables to be generated at once. This is done by defining a new function (such as FIGURES) that is processed by the same subroutine (CONTENTS) as the CONTENTS function. The LOAD control phrase is used to do that definition:

```
COMMENT ... DEFINE CONTENTS FUNCTION
LOAD CONTENTS
COMMENT ... DEFINE THE FIGURES FUNCTION
LOAD FIGURES * CONTENTS
```

Once the FIGURES function has been defined, it is used exactly as the CONTENTS function is. As many different tables as desired may be in progress at once by issuing the appropriate LOAD control phrases.

Since the table of contents is actually printed at the end of the document (it will have to be moved to the beginning after printing), it is necessary to reset the page number to 1 before printing the table of contents. If the page numbers are to be printed as Roman numerals, this could be done by:

```
...
|CONTENTS(OFF)
|PAGE(ROMAN,1)
|HEADING('TABLE OF CONTENTS')
|CONTENTS(INITIALIZE)
|CONTENTS(GENERATE)
```

See the description of CONTENTS in the section "FORMAT Functions" for complete details on this function.

PRODUCING AN INDEX

An index may be easily generated for a document using the INDEX function. When INDEX is called to make an entry into the index, it remembers the entry and the current page number. At the end of the document, INDEX is called again to actually print the index. The index is automatically sorted and formatted.

This section introduces the INDEX function (see the section "FORMAT Functions" for complete details). For example,

```
The FORMAT functions |INDEX('functions') are
...
An index |INDEX('functions, index') |INDEX('index') may be
...
Table of contents |INDEX('functions, contents') |INDEX('contents')
...
Multiple tables |INDEX('functions, contents') are produced by
...
/S/|INDEX(init) |INDEX(gene)/L/
```

produces the index

```
Contents ..............................................5
Functions .............................................2
  Contents ........................................5-8
  Index ...............................................7
Index .................................................7
```

INDEX uses commas to indicate subentries. It is possible to produce an index where page numbers appear immediately after the items, as well as multiple-column indices. The capitalization of entries may also be altered.

Making an Index

An index entry is created by calling INDEX at the point in the text to which the entry is to refer. The simplest form of this call is

```
|INDEX('entry')
```

where "entry" is the index entry, enclosed in primes. If "entry" contains a comma, the string to the right of the comma is taken as a subentry of the string on the left. Note the call |INDEX('functions, index') above. There are also three optional arguments to specify a prefix and postfix to be placed on the entry after the index is alphabetized, and to specify that more than one subentry level may be used.

After all entries have been made, the index is actually generated by two calls. The first

```
|INDEX(INITIALIZE)
```

generates a series of control phrases to set tabs and indents and define a set of macros. Then, a call of the form

```
|INDEX(GENERATE)
```

sorts and prints the index. If it is desired to start a new page or print a heading for the index, that should be done before this last call is made.

Altering the Format

Often it is desirable to put a "heading" into the index which does not have a page number associated with it. The call

```
|INDEX(NOPAGE)
```

omits the page number on all subsequent entries. The call

```
|INDEX(PAGE)
```

reverses this. For instance, if the first line of the example at the beginning of this section had read

```
The FORMAT functions |INDEX(NOPAGE) |INDEX('functions')
|INDEX(PAGE)
```

the index would have been

```
Contents .....................................5

Functions
  Contents ...................................5-8
  Index ......................................7
Index ........................................7
```

Instead of the dots and right-aligned page numbers which normally appear in the index, it is possible to have the page numbers printed immediately after the entries. The call

```
|INDEX(COLUMN)
```

generates the entries in this format. With this option, the index at the beginning of this section would appear as

Other options are available (see the section "FORMAT Functions" for further details).

Making Several Indices Simultaneously

It is possible to have more than one index for a document by loading more than one INDEX function. The INDEX function is loaded automatically at first reference; another copy can be loaded by specifically loading it under another name with the control phrase

```
LOAD name * INDEX
```

where "name" is the name to be used for the new copy. After this is done, "name" may be called just as INDEX is, but the index produced are distinct from that produced by calls to INDEX.

April 1977

FORMAT FUNCTIONS

FORMAT provides a set of predefined functions, called FORMAT functions. These are:

```
BLOCK    - produces block letters
CENTER   - centers a string in a field
CONTENTS - generates a table of contents
COUNT    - provides counters
DATE     - prints the date/time in various formats
EXIST    - checks the existence of a macro
FOR      - repeats a string a number of times
INDEX    - generates an index
INPUT    - reads input from GUSER
JUST     - justifies a string in a field
LIBRARY  - identifies a macro library
MOST     - loads other functions
NULL     - checks whether a macro parameter is null
PAGE     - provides more flexible page numbering
SPREAD   - spaces out the characters of a string
STACK    - provides a stack for some FORMAT variables
```

All functions in FORMAT, including user-defined functions, may be called in text mode by either

|function(parlist)␣

or

|function(parlist)|

where "␣" represents a blank. An "!" may be used in place of the "|" in the above forms. The second form requires no blanks and thus may appear in the middle of a word. A function may also be called in control phrase mode by

CALL function(parlist)

where "function" is the name of the function to be called and "parlist" the parameter list to be passed to the function. If there are no parameters, "parlist" and the enclosing parentheses may be omitted. Parameters in "parlist" must be separated by commas. If parameters contains blanks or commas, that parameter generally must be enclosed in primes. Certain functions require that some parameters always be enclosed in primes; such cases are noted in the descriptions. "parlist" should not contain blanks except between primes. A FORMAT function is loaded automatically upon first reference.

Occasionally, it is useful to have a copy of a FORMAT function loaded under an "alias," so that, for instance, two separate copies of the function could be in use at the same time. This may be accomplished with the LOAD control phrase.

When the functions are processed by FORMAT, the resulting expanded text (if any) is placed into the input stream. This expanded text then processed by FORMAT for inclusion into the generated output.

BLOCK

The BLOCK function generates block letters similar to those used on MTS job tailsheets. This is often useful for title pages.

Letters are 12 characters by 12 characters, with a 2-character space between letters. The calling sequence is

```
|BLOCK(string)
```

where "string" is the character string to be block-lettered. The string should not be enclosed in primes, and may contain at most 9 characters. Block letters are available only for letters and numbers. For example,

```
|BLOCK(ABC)
```

produces

```
 AAAAAAAAAA   BBBBBBBBBBB    CCCCCCCCCC
AAAAAAAAAAAA  BBBBBBBBBBBB  CCCCCCCCCCCC
AA        AA  BB        BB  CC        CC
AA        AA  BB        BB  CC
AA        AA  BB       BB   CC
AAAAAAAAAAAA  BBBBBBBBBB    CC
AAAAAAAAAAAA  BBBBBBBBBB    CC
AA        AA  BB       BB   CC
AA        AA  BB        BB  CC
AA        AA  BB        BB  CC        CC
AA        AA  BBBBBBBBBBBB  CCCCCCCCCCCC
AA        AA  BBBBBBBBBBB    CCCCCCCCCC
```

CENTER

The CENTER function centers a character string within a field. This may be useful for constructing tables.

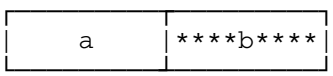
CENTER takes as arguments a character string, the length of field in which to center it, and the fill character to be used in filling the remainder of the field. The calling sequence is:

```
|CENTER(string,length,fill)
```

where "string" is the character string to be centered, enclosed in primes. Alternatively, if CENTER is called from within a macro, it may be the number of a parameter of that macro. In this case, the parameter is taken as the string to be centered. "length" is the length of the field in which to center "string". If "length" is omitted, the last value specified in a call is used. "fill" is the fill character, entered as a single character or a two-character hexadecimal number. "fill" defaults to the nontrivial blank. For example,

```
BOX 1 20 30 40
TAB 21
GO
/B1LT/ |CENTER('A',9) |CENTER('B',9,*) /LB1L/
```

produces



CONTENTS

The CONTENTS function is used to produce a table of contents or similar tables of figures or other material in a document. Calls to CONTENTS can be classified into three types. Calls of the first type are those calls that produce entries into the table of contents. In this type of call, the actual entry and, optionally, a level, prefix, and postfix are specified. There is also a call to ignore subsequent entries, and a call to resume making entries. In the second type of call, options which affect the format of the table are specified. Mainly, this involves the way in which the page number is printed. Calls of the third type are those calls which actually generate the table. These calls are made at the end of the document.

Producing Entries in the Table

An entry into the table is produced by a call of the form

```
|CONTENTS(entry,level,prefix,postfix)
```

"entry" is the character string which is to appear in the table (it must be enclosed in primes). "level" is the level of the entry. Successive levels are indented in the table. The allowable levels are 1 through 11; 1 is the main level; "level" defaults to 1. "prefix" and "postfix" are optional arguments. They specify character strings to precede and follow, respectively, "entry". "prefix" and "postfix" should be enclosed in primes. CONTENTS does not produce any blanks between them and "entry". These two arguments are most useful for including section numbers, etc., which are generated by counters or macros.

CONTENTS ignores all calls of the above form if it is called with

```
|CONTENTS(OFF)
```

A call of the form

```
|CONTENTS(ON)
```

may be used to resume making entries into the table after a call to |CONTENTS(OFF).

Altering the Format of the Table

For all calls of this type, only the entries made to the table of contents subsequent to the call are affected.

Normally, the page numbers in the table of contents are left-justified in a field. Right-justified entries are produced if a call is made in the form

```
|CONTENTS(PAD,length,fill)
```

"length" is the length of the field to use. It should be large enough to accommodate the longest page number. "fill" is the fill character enclosed in primes or a two-digit hexadecimal number. "fill" is optional and defaults to the nontrivial blank.

There are several calls which affect whether the page number is printed as an integer, Roman numeral, etc. By default, integer page numbers are printed in the table of contents. If, for example, the PAGE function has been called to produce Roman numeral page numbers or insert pages, the page numbers which appear in the table of contents will be of a different form than those which appear on the pages. To remedy this

```
|CONTENTS(REAL)
```

may be used. If this is used, the page numbers in the table of contents will be in exactly the same form as those on the text pages. The default is equivalent to

```
|PAGE(VIRTUAL)
```

If the PAGE function is not used, the default will produce the same results as |CONTENTS(REAL). The form in which the page number is printed (Roman, alphabetic, etc.) may be changed by

```
|CONTENTS(type)
```

where "type" is one of ROMAN, ALPHA, NAME, and INTEGER. These types have the same meaning as for the PAGE function. If a call of this type is made, page numbers in the table of contents entries are printed in the form specified, regardless of how they appear in the document.

Normally, if the table of contents contains entries which do not fit on a single line, those entries will be indented from the right so that they do not overlap the page number field; lines after the first also will be indented from the left. For example,

```
Here Is The First Table Of Contents
    Entry ..........................1
```

If this indentation is not desired,

```
|CONTENTS(SHORT)
```

should be used. The indentation is accomplished with the INDENTS control phrase and /I/. These indents are set if (1) at least one "too long" entry occurs and |CONTENTS(SHORT) is not used, or (2)

```
|CONTENTS(LONG)
```

is used.

Printing the Table

Two calls are required to actually generate the table of contents after all entries have been made. The first call defines some macros and defines new tab and indent settings after saving the current ones. The second call actually prints the table. If the user wishes to change, for instance, the indentations used in the table, he may do so by changing them between the two calls.

The first call has the form

```
|CONTENTS(INITIALIZE,width,prefix,postfix)
```

"width" is an optional argument. If present, it gives the width (as in the WIDTH control phrase) to be used. If it is omitted, the current width is used. "prefix" and "postfix" are optional character strings which must be enclosed in primes. "prefix" is placed immediately in front of each entry, if given, and "postfix" is placed immediately after each entry, if given. If "prefix" and "postfix" are both '/@/', for instance, all entries appear in uppercase.

The table is actually printed by

```
|CONTENTS(GENERATE)
```

If the table is to have a heading, or the page number is to be reset, it should be done before this call.

Examples

The section "Producing a Table of Contents" contains elementary examples. The examples here are more concerned with the fine points.

The first example illustrates the effects of REAL, VIRTUAL, and the page number types. The FORMAT input

```
CALL PAGE(ROMAN)
GO
THIS IS PAGE 1.  |CONTENTS('PAGE 1 ENTRY')
/S/THIS IS PAGE 2.  |CONTENTS('PAGE 2 ENTRY') |CONTENTS(REAL)
|CONTENTS('PAGE 2, REAL')
/S/THIS IS PAGE 3.  |CONTENTS('PAGE 3, REAL')
|CONTENTS(VIRTUAL)
|CONTENTS('PAGE 3, VIRTUAL') /S/THIS IS PAGE 4.  |CONTENTS(ALPHA)
|CONTENTS('PAGE 4, VIRTUAL, ALPHA') |CONTENTS(REAL)
|CONTENTS('PAGE 4, ALPHA, REAL')
/S/ |CONTENTS(INIT) |CONTENTS(GENE)/L/
```

produces (table of contents portion only)

The second example shows the use of the prefix, postfix, and level in entries. The input

```
        DEF HEAD
        ||CONTENTS('|PAR.1|',1,'|LHEAD|.|LSUB|/@/¬','/@/')
        |/V/
        |SET LSUB = '1'
        |GO
        |/L2@/|PAR.1|/@/
        DEF SUBHEAD
        ||CONTENTS('|PAR.1|',2,'|LHEAD|.|LSUB|¬')
        |/L/|PAR.1|
|       SET LHEAD = '1'
        SET LSUB = '1'
        GO
        |HEAD('FIRST SECTION') |SUBHEAD('FIRST, SUB 1')
        /P/THIS IS SECTION 1, SUB 1
        /V/
        SET LSUB = '2'
        GO
        |SUBHEAD('FIRST, SUB 2') /P/ THIS IS SECTION 1, SUB 2.
        /V/
        SET LHEAD = '2'
        GO
        |HEAD('SECOND SECTION') |SUBHEAD('SECOND, SUB 1')
        /P/THIS IS SECTION 2, SUB 1.
        /S/ |CONTENTS(INIT)  |CONTENTS(GENE) /L/
```

produces (table of contents portion only):

```
 1.1 FIRST SECTION ..........................................2
   1.1 first, Sub 1 .......................................2
   1.2 first, Sub 2 .......................................3
 2.1 SECOND SECTION ........................................5
   2.1 second, Sub 1 ......................................6
```

COUNT

The COUNT function is a generalization of the counter special operator. It provides a larger number of counters, more choice in the way counters are printed, a means for referring to a counter without incrementing it, and a settable increment.

There are three types of calls to the COUNT function. The first, to
| |COUNT, initializes a counter. This call should be made for a counter
| before it is first used. The second type, all other calls to |COUNT,
| prints the value of the counter and then increments it. The third, to
|COUNTV, prints the value of a counter without incrementing it.

One hundred counters are available for use, numbered 1 to 100. They are distinct from the counters used by the counter special operator.

Counters may take any value that will fit in a fullword integer. Counters may also be referred to by 1- to 8-character symbolic names. To use a symbolic name, the name is used instead of a number when initializing the counter. A counter number is assigned by COUNT; assignments are made beginning with counter 100 for the first symbolic name initialized and counting downwards.

A call to initialize a counter takes the form

```
|COUNT(counter,first,inc,type,length,fill)
```

"counter" is the name (1-8 characters) or number (1-100) of the counter being initialized. "first" is the integer first value the counter will take. "inc" is the integer increment between successive values; "inc" defaults to 1. "type" is the form in which the counter is printed. The default is INTEGER. The available types are:

INTEGER	printed as integers; any fullword integer value is printed
ROMAN	printed as lowercase Roman numerals; only values 1-99 are printed
ALPHA	printed as lowercase letters a,b,...,aa,...,zz; only values 1-702 are printed
NAME	printed as lowercase spelled-out names; only values 1-99 are printed
LINE	printed as MTS line numbers (value is divided by 1000 and printed with 3 decimal places); all fullword integer values are printed
OCTAL	printed as octal numbers, unsigned and with no leading zeros
SUPERSCRIPT	as INTEGER, but printed as superscript
NUMERIC	synonym of INTEGER
DUMMY	value is not printed (incrementing is done)

CAP may be appended to ROMAN, ALPHA, or NAME to produce uppercase. "length" and "fill" are optional; they are used to produce fixed-length output. "length" is the length of the field in which the value is to be placed. "fill" is the pad character used to fill the remaining space in the field, if any. If "length" is positive, padding with "fill" is done on the left; if "length" is negative, padding is done on the right. The default "fill" is "¬" (nontrivial blank). "fill" should be a single character in primes or a two-digit hexadecimal number.

"inc", "type", "length", or "fill" may be changed for a counter without reinitializing it by omitting "first". For instance, if counter 2 has been initialized, its increment can be changed to 3 and its type to superscript with

```
|COUNT(2,,3,SUPERSCRIPT)
```

A counter is referenced with a call of the form

```
|COUNT(counter)
```

where "counter" is the number or name of the counter being referenced. When this call is made, the value of the counter is printed in the manner specified in the initialization call, and then the counter is incremented.

The third form of the call prints the counter without incrementing it:

```
|COUNTV(counter)
```

"counter" is the number or name of the counter. The value is placed into the FORMAT input stream as specified on the last initialization call. The value is the same value printed on the last call to |COUNT (counter). For example,

```
TABS 10 20 30
GO
|COUNT(1,10,3,INTEGER)
|COUNT(2,98,2,ROMAN)
|COUNT(ABC,24,1,ALPHA,2,'.')
/LT/|COUNT(1)/T/|COUNT(2)/T/|COUNT(ABC) /L/
/T/|COUNT(1)/T/|COUNT(2)/T/|COUNT(ABC) /L/
/T/|COUNTV(1)/T/|COUNTV(2)/T/|COUNTV(ABC) /L/
|COUNT(1,,,ROMAN) |COUNT(2,,,INTEGER)
/LT/|COUNT(1)/T/|COUNT(2)/T/|COUNT(ABC) /L/
/T/|COUNT(1)/T/|COUNT(2)/T/|COUNT(ABC) /L/
/T/|COUNTV(1)/T/|COUNTV(2)/T/|COUNTV(ABC) /L/
```

produces

```
10      xcviii      .x
13                  .y
13      1           .y
xvi     102         .z
xix     104         aa
xix     104         aa
```

DATE

The DATE function allows the printing of the current date and/or time in a variety of formats.

DATE is called with a character string into which parts of the date are to be inserted and a list of the parts of the date which are to be inserted. The calling sequence is

```
|DATE(string,list)
```

"string" is the character string into which parts of the date are to be placed; "string" must be enclosed in primes. An * should appear wherever some part of the date is to be placed. "list" is one or more arguments specifying the parts of the date to be inserted, in the order in which they are to appear in "string". Each must be one of the following:

YEAR	4-digit number of the current year
YY	last 2 digits of the current year
MONTH	name (given in lowercase) of the current month
MM	2-digit number of the current month
WEEKDAY	name (given in lowercase) of the current day of the week
DAY	1- or 2-digit current day of the month; no leading zero
DD	2-digit day of the month
TIME	current time on 12-hour clock, in the form hh:mm a.m. (or p.m.)
TT	current time on 24-hour clock in the form hh:mm
HOURS	1-or 2-digit current hour on 12 hour clock; no leading zero
HH	2-digit current hour on 24-hour clock
MINUTES	1- or 2-digit current minute; no leading zero
MN	2-digit current minute
SECONDS	1- or 2-digit current second; no leading zero
SS	2-digit current second
AM or PM	"a.m." or "p.m.", as appropriate for the current time

If DATE is called with no arguments, it behaves as if it were called with

```
|DATE(' * @* * *',YEAR,MONTH,DAY,TIME)
```

For example:

Input	Output
`|DATE('* @* * *',YEAR,MONTH,DAY,TIME)`	1977 January 12 10:77 p.m.
`|DATE('*-*-**:*:*',YY,MM,DD,HH,MN,SS)`	77-01-1222:17:31
`|DATE(' *, @*',TT,WEEKDAY)`	22:17, Wednesday
`|DATE('*:*:* *',HOURS,MINUTES,SECONDS,AM)`	10:17:46 p.m.

EXIST

The EXIST function returns one of two character strings passed to it, depending on whether a macro has been defined.

The calling sequence is

```
|EXIST(macro,def,notdef)
```

"macro" is the name of a macro or function whose existence is to be tested for. A macro is defined with the DEFINE control phrase. A function is "defined" by loading it using the LOAD control phrase, or, in the case of a FORMAT function, referring to it. "def" and "notdef" are character strings; either may be null. If "def" or "notdef" contains blanks or commas, it must be enclosed in primes. If "macro" is defined, "def" is placed in the FORMAT input stream. Otherwise, "notdef" is placed in the FORMAT input stream. For example,

```
DEFINE A
|A MACRO
GO
|EXIST(A,abc,def) /L/
|EXIST(C,abc,def) /L/
```

produces

```
abc
def
```

FOR

The FOR function puts a specified number of copies of a character string into the FORMAT input stream, optionally placing the value of a "loop counter" into each string.

The calling sequence is

```
|FOR(delim,string,first,last,inc)
```

"string" is the character string to be repeated. If "string" contains blanks or commas, it must be enclosed in primes; otherwise primes are optional. "delim" is a two-character string. The second character is the terminating delimiter for "string". It must appear in "string", and only the characters in "string" up to the first instance of this character are repeated. All instances in "string" of the first character in "delim" are replaced, in each repetition, by the current value of the "loop counter". "first", "last", and "inc" are treated as in a DO(FOR) loop. All three are integers. The "loop counter" is given the value "first". At each repetition, the current value of the "loop counter" is inserted wherever necessary into "string", a copy of "string" is placed into the FORMAT input stream and "inc" is added to "loop counter". Repetitions continue until "loop counter" is greater than "last" (for "inc" positive), or "loop counter" is less than "last" (for "inc" negative). "inc" is optional. If "inc" is not specified, it defaults to 1 if "first" is less than or equal to "last", and -1 otherwise. For example,

```
|FOR(*%,' /L/ |33* = |*%',32,36) /L/
```

produces

```
|32 = &
|33 = |
|34 = ¬
|35 = <
|36 = =
```

INDEX

The INDEX function is used to produce an index for a document.

Calls to INDEX can be classified into three types. Calls of the first type are those which enter information into the index. In this type of call the entry, and, optionally a maximum subentry level and a prefix and a postfix are specified. There is also a call to ignore subsequent entries, and one to resume making entries. In the second type of call, options which affect the format of the index are specified. Calls of the third type are those which actually generate the index.

Making Index Entries

An entry in the index is produced by a call of the form

```
|INDEX(entry,level,prefix,postfix)
```

"entry" is the character string which is to appear in the index. It must be enclosed in primes. Sublevels in the entry should be indicated by commas. For example,

```
|INDEX('functions,contents') |INDEX('functions,index')
```

produces the index entries

```
Functions, Contents .....................................10
  Index .................................................10
```

If the level of the subentry is greater than 2 (the example above has level 2), the optional parameter "level" should be used. "level" specifies the maximum level which may be used for the subentry. The actual level used is determined by the commas in "entry", but a level no higher than "level" will be used. "level" defaults to 2. "prefix" and "postfix" are optional character strings which are to be concatenated as prefix and postfix, respectively, to "entry" after the index has been alphabetized. Typically, "prefix" and "postfix" consist of command operands (especially capitalization, since the case of a letter affects

its position in sorting). When these three optional arguments are used frequently, it is often convenient to have the actual call to INDEX generated by a macro.

INDEX ignores all calls of the above form which occur after the call

```
|INDEX(OFF)
```

The call

```
|INDEX(ON)
```

is used to resume making entries after a call to |INDEX(OFF). The most common use of INDEX(OFF) is to prevent an index entry from being made by a macro which produces headings when that macro is being called to generate a heading for the index.

Altering the Format of the Index

INDEX has calls to set (1) whether the page number is printed for an entry, (2) the way page numbers are printed--using Roman numerals or letters (3) whether page numbers are printed immediately after the entry or right-aligned, and (4) the spacing of entries.

• PAGE, NOPAGE

If INDEX is called with

```
|INDEX(NOPAGE)
```

the page number is omitted from subsequent entries, until a call

```
|INDEX(PAGE)
```

is made. This is useful for generating headings in an index. For example,

```
FORMAT functions |INDEX(NOPAGE) |INDEX('functions')
|INDEX(PAGE)
...
The PAGE function |INDEX('functions, page')
...
The INDEX function |INDEX('functions,index')
```

produces the index entries

```
Functions
  Index ..................................................23
  Page ...................................................10
```

• RANGING, NORANGING

Normally, a range of page numbers is indicated by printing the first and last page numbers with a "-" between them. If

```
|INDEX(NORANGING)
```

is given, every page reference is printed separately. The call

```
|INDEX(RANGING)
```

restores the normal behavior. For example,

```
|INDEX('functions,page') /S/ |INDEX('functions,page')
|INDEX(NORANGE)
...
|INDEX('functions,index') /S/ |INDEX('functions,index')
```

produces the index

```
Functions, Index .....................................12,13
  Page .....................................................11-12
```

• ROMAN, ALPHA, INTEGER

The default type of page numbering is integer. This may be changed by a call of the form

```
|INDEX(type)
```

where "type" is one of ROMAN, ALPHA, or INTEGER. These have exactly the same meaning as for the page function.

• PAD

If

```
|INDEX(PAD,length,char)
```

is used, page numbers are printed right-aligned in a field of length "length", padded on the left with "char". "char" is optional; the default is ".". "length" defaults to 10. For example,

```
|INDEX(PAD,3,#)
|INDEX('functions,index')
```

produces the index

```
Functions, Index ....................................#13
```

• COLUMN

Page numbers are printed immediately after the entries, instead of right-aligned, if

```
|INDEX(COLUMN)
```

is used. For example,

```
|INDEX(COLUMN) |INDEX('functions,index')
|INDEX('functions,page')
```

produces the index entries

```
Functions, Index 16
  Page 16
```

The default format is

```
|INDEX(TAB)
```

For example,

```
|INDEX(TAB) |INDEX('functions,index')
|INDEX('functions,page')
```

produces

```
Functions, Index .....................................16
  Page .....................................................16
```

Note that this is the default, and |INDEX(TAB) need be used only to reset after the format has been changed.

• LIST

The call

```
|INDEX(LIST)
```

causes page numbers to be omitted entirely.

• SPACER

If there is a list of page numbers for an entry, the page numbers are placed on one line, separated by commas, as in

```
Functions, Index ....................................18,20
```

The separator character may be changed. The call

```
|INDEX(SPACER,string)
```

causes the commas to be replaced by "string". "string" consists of one or more characters, and must be enclosed in primes. If

```
|INDEX(SPACER,';')
```

is used, the above is printed as

```
Functions, Index .....................................18;20
```

• SKIP

The call

```
|INDEX(SKIP)
```

causes INDEX to leave a blank line each time a new letter begins a first level entry.

Printing the Index

Two calls are required to actually generate the index after all entries have been made. The first call defines macros, and defines new tab and indent settings after saving the current ones. The second sorts and prints the index. If the user wishes to change the number or width of columns, it must be done before the first call; changing the tabs or indents used for printing the index must be done between the two calls.

The first call has the form

```
|INDEX(INITIALIZE,width,prefix,postfix,wsort)
```

Only the first parameter is required. "width" is the optional width to use (as in the WIDTH control phrase). If it is omitted, the current width is used. "prefix" and "postfix" are character strings to be placed before and after, respectively, each entry after sorting (alphabetizing). They may include command operands; for instance, if both are /@/, all entries are printed in uppercase. "wsort" is the length of sorting field passed to the SORT system subroutine. "wsort" defaults to 50, which means that the only first 50 characters of each entry are considered for alphabetizing. This is usually adequate.

The index is actually printed by

```
|INDEX(GENERATE)
```

If the index is to have a heading, or a new page started, it must be done before the call.

For example, the following input

```
LINES 11
GO
THIS IS PAGE 1. |INDEX(NOPAGE) |INDEX('FUNCTIONS') |INDEX(PAGE)
|INDEX('FUNCTIONS, CONTENTS') |INDEX('CONTENTS')
/S/THIS IS PAGE 2. |INDEX('FUNCTIONS, CONTENTS') |INDEX('CONTENTS')
|INDEX('FUNCTIONS, PAGE') |INDEX('PAGE')
/S/THIS IS PAGE 3. |INDEX('FUNCTIONS, INDEX') |INDEX('INDEX')
/S/THIS IS PAGE 4. |INDEX('FUNCTIONS, CONTENTS') |INDEX('CONTENTS')
/S/THIS IS PAGE 5. |INDEX('FUNCTIONS, ALL') |INDEX('ALL')
/SM@/ INDEX /@MLV/
BETWEEN 5
COLUMN 2
GO
|INDEX(COLUMN) |INDEX(INIT) |INDEX(GENERATE) /S/
```

produces the index

```
                                        INDEX
          All 5                           Index 3
          Contents 1-2,4                  Page 2
          Functions                     Index 3
            All 5                       Page 2
            Contents 1-2,4
```

INPUT

The INPUT function reads input from the logical I/O unit GUSER and places it in the FORMAT input stream. FORMAT reads from the logical I/O unit SCARDS, so this provides a way to include input from an alternate source.

The calling sequence is

```
|INPUT(prompt,prefix,oneof,onnull)
```

All four parameters are optional. Each call to INPUT reads a line from GUSER. "prompt" is a character string which should be enclosed in quotes, if it contains blanks or commas. If it is present, it is written on the logical I/O unit SERCOM as a prompting message before reading input. "prompt" and "prefix" are generally used only when reading from a terminal. "prefix", if present, is used as a prefix character on the call to GUSER. "oneof" is a string to be returned if an end-of-file occurs on GUSER. If "oneof" is omitted, no string is returned when an end-of-file occurs. "onnull" is a string to be returned if the line read from GUSER has zero length. If "onnull" is omitted, no string is returned. "oneof" and "onnull" must be enclosed in primes, if they contain blanks or commas.

For example, the following FORMAT source

```
effective date:  |INPUT(effective,?,'|D','|D')
```

might be used, if FORMAT is being run from a terminal, to print "effective" on the terminal, to read the user-supplied date from GUSER using "?" as the prefix character, and to use the current date, if a null string or end-of-file is entered.

JUST

The JUST function right- or left-aligns a character string within a field of specified length. This is useful for constructing tables.

JUST takes as parameters a character string, the length of the field, and a character to be used to fill the rest of the field. The function places in the FORMAT input stream a string whose length is the field length, with the character string right-aligned or left-aligned in it and the rest filled with the pad characters.

The calling sequence is

```
|JUST(string,length,pad)
```

"string" is the character string; it must be enclosed in primes. Alternatively, if JUST is called from within a macro, "string" can be the number of a parameter to that macro; the macro parameter is taken as the character string. In either case, any command operands or special operators in the character string are processed after the string is right-aligned (hence the alignment will be off). "length" is the integer length of the field in which the string is to be right-aligned. If "length" is negative, the character string is left-aligned instead. If "length" is omitted, the last specified value for "length" is used. "pad" is the pad character; if it is omitted, a nontrivial blank is used.

For example,

```
BOX 1 1 11 23
GO
/B1L/¬|JUST('LEFT1',-10) |JUST('RIGHT1',10)
/L/¬|JUST('LEFT2',-10) |JUST('RIGHT2',10)/LB1L/
```

produces

```
┌─────────┬──────────┐
|left1    |     right1|
|left2    |     right2|
└─────────┴──────────┘
```

LIBRARY

The LIBRARY function specifies the location of a macro library.

The calling sequence is

|LIBRARY(fdname)

where "fdname" is the name of the file containing the macro library. Once LIBRARY has been called with "fdname", the macro library contained in "fdname" is searched whenever a macro invocation is encountered for a macro which has not yet been defined. If the macro definition is in the library, the macro will be defined, and an entry made in the control phrase listing. All macro libraries for which calls to LIBRARY have been made will be searched for "fdname".

A macro library is a line file containing a directory of the macro names and their macro definitions. Each directory entry contains the name of a macro starting in column 1 followed by at least one blank, followed by the line number of the DEFINE command for the macro (only positive, integral line numbers may be used). The line following the last directory entry contains a blank in column 1. The first macro definition follows the end of the directory. The first line of a definition is a DEFINE command (uppercase, not abbreviated). Subsequent lines contain the definition of the macro in the usual format, each line beginning with the character "|". For example,

Line #	Line Text
1	MACRO1 5
2	MACRO2 9 (directory entries)
3	MACRO3 13
4	
5	DEFINE MACRO1
6	|xxxxxxx
7	|xxxxxxx (macro definition)
8	|xxxxxxx
9	DEFINE MACRO2
10	|xxxxxxx
11	|xxxxxxx (macro definition)
12	|xxxxxxx
13	DEFINE MACRO3
14	|xxxxxxx
15	|xxxxxxx (macro definition)
16	|xxxxxxx

NULL

The NULL function is intended to be called from within a macro. It returns one of two character strings passed to it, depending on whether a parameter to the macro is present.

The calling sequence is

```
|NULL(par,null,notnull)
```

"par" is the number of the parameter in the macro to be checked. It may appear either as a number, or as '|PAR.n', where "n" is the parameter number. The primes are required in the second form. "null" and "notnull" are character strings, which must be enclosed in primes if they contain blanks or commas. If "par" is not present, i.e., is null, in the macro call, string "null" is returned to the FORMAT input stream. Otherwise, "notnull" is returned. For example,

```
DEF A
||NULL(1,ABC,DEF) |NULL('|PAR.1',ABC,DEF)
GO
|A /L/ |A(X) /L/
```

produces

```
abc abc
def def
```

PAGE

The PAGE function provides more flexibility in page numbering. It can be used to print Roman numerals, alphabetic, or character page numbers, to print the page number in the text, or to skip page numbers.

There are several calling sequences. For each, the first argument specifies the option to be used. This is one of the character strings INTEGER, ROMAN, NAME, ALPHA, RESET, ADD, SAVE, SET, CONTINUE, VALUE, INSERT, NORMAL, IF, ON, or OFF. Only the first four characters are required. The remaining arguments specify particular information for that option.

Each time the page number is printed, it is incremented and the new value is stored to be used the next time. (Note that the page number stored is changed at a different time when it is printed at the bottom of the page than when it is printed at the top.) Thus, FORMAT always stores the next page number to be used. The SAVE and ADD options operate on this next page number.

Descriptions of the options follow.

• Setting the page number and increment

|PAGE(SET,page,inc,n)

The next page number to be used will be page+n. If "n" is omitted, n=0 is assumed. "inc" becomes the increment between successive page numbers. If "inc" is omitted, inc=1 is assumed. "page", "inc", and "n" are integers. For example, |PAGE(SET,29,2) sets the next page number to 29, the page following that will have number 31.

• Skipping pages

|PAGE(ADD,n)

The next page number to be used will be "n" plus the current page number. "n" is an integer. For example, if |PAGE(ADD,5) occurs on page 23, the next page number used will be 29.

• Roman numerals, etc.

|PAGE(type,page)

"page" is the next page number to use. If it is omitted, numbering continues normally. "type" is one of INTEGER, ROMAN, ALPHA, NAME. INTEGER is the default type. ROMAN produces lowercase Roman numerals. NAME produces page numbers spelled out in lowercase (e.g., nineteen). ALPHA produces page letters (e.g., f for 6, aa for 27). Numbering continues with the type specified until a call to PAGE is made specifying a different type.

Page Revised February 1979 April 1977

• Printing the page number

|PAGE(VALUE,string,delim)

The current page number is returned to the FORMAT input stream embedded in "string" at the first occurrence of the character "delim". "string" and "delim" do not need to be enclosed in quotes unless they contain blanks or commas. For example,

|PAGE(VALUE,*****,+)

would put **17** into the text if the current page number were 17.

• Fractional page numbers

|PAGE(INSERT)

This numbers subsequent pages with decimal numbers from the current page. For example, if the current page is 14, a |PAGE(INSERT) causes the following pages to be numbered as 14.1, 14.2, ..., 14.10, 14.11, ...

|PAGE(NORMAL)

This resets page numbering to normal after an |PAGE(INSERT).

• Enabling and disabling numbering

|PAGE(OFF)

This disables page numbering. Pages continue to be counted, but the page numbers are not printed.

|PAGE(ON)

This enables page numbering.

• Saving the page number and resetting it to the value saved

|PAGE(SAVE,page)

The next page number to be used is saved by PAGE for use by |PAGE(RESET). "page" becomes the next page number used. "page" should be an integer.

|PAGE(RESET,n)

The page number is reset to "n" plus the value saved by the last |PAGE(SAVE). For example,

```
|PAGE(SAVE) text/S/ more text |PAGE(RESET)
/S/still more text
```

results in the pages containing "more text" and "still more text" having the same page number.

• Resetting the page number to the largest used so far

```
|PAGE(CONTINUE,n)
```

The page number is reset to its largest value so far plus "n".

• Executing a page option conditional on the current page being right or left

```
|PAGE(IF,cond,opt,...)
```

"cond" is EVEN, ODD, RIGHT, or LEFT (EVEN and ODD refer to the current page number, RIGHT and LEFT refer to right and left page). If the condition is true, the PAGE function behaves as if it were called with the parameters "opt,..." For example, |PAGE(IF,EVEN, ADD,1) will skip the next page number, if it is even.

The following example shows the use of the PAGE function to print page numbers as Roman numerals. The following input

```
WIDTH 60
TEXT 2
LINES 8
LOWERCASE
FOOTER
/J/ ¬ /E/
PAGE 3 8 31
CALL PAGE(ROMAN)
GO
Dixon broke into a frenzied, lung-igniting sprint, while the
conductor watched him immobile from the platform.  When he
was halfway to the bus, this person rang the bell, the driver let
in the clutch, and the wheels began to turn.  Dixon found he was
even better at running than he'd thought, but when the gap
between man and bus had narrowed to perhaps five yards, it began
to widen rapidly.  Dixon stopped running and favoured the
conductor, who was still watching unemotionally,
with the best-known
obscene gesture.  At once the conductor rang the bell again and
the bus stopped abruptly.  Dixon hesitated for a moment, then
trotted lightly up to the bus and boarded it with some diffidence.
He found himself unwilling to meet the eye of the conductor, who
now said admiringly, "Well run, whacker," and rang the bell
for the third time.
```

produces as output

```
|-------------------------------------------------------------|
|                                                             |
| Dixon broke into a frenzied, lung-igniting sprint, while the |
| conductor watched him immobile from the platform. When he |
| was halfway to the bus, this person rang the bell, the |
| driver let in the clutch, and the wheels began to turn. |
| Dixon found he was even better at running than he'd thought, |
|                                                             |
|                          iii                                |
|-------------------------------------------------------------|

|-------------------------------------------------------------|
|                                                             |
| but when the gap between man and bus had narrowed to perhaps |
| five yards, it began to widen rapidly. Dixon stopped running |
| and favoured the conductor, who was still watching |
| unemotionally, with the best-known obscene gesture. At once |
| the conductor rang the bell again and the bus stopped |
|                                                             |
|                          iv                                 |
|-------------------------------------------------------------|

|-------------------------------------------------------------|
|                                                             |
| abruptly. Dixon hesitated for a moment, then trotted lightly |
| up to the bus and boarded it with some diffidence. He found |
| himself unwilling to meet the eye of the conductor, who now |
| said admiringly, "Well run, whacker," and rang the bell for |
| the third time.                                             |
|                                                             |
|                           v                                 |
|-------------------------------------------------------------|
```

SPREAD

The SPREAD function takes a character string and separates the characters on it by nontrivial blanks. This is often useful in titles.

The calling sequence is

```
|SPREAD(string,break)
```

"string" is the character string into which blanks are to be inserted. It should be enclosed in primes if it contains blanks or commas (otherwise, primes are optional). "break" is a string of one or more characters to be inserted between the characters of "string". "break" is optional; if it is omitted, a single nontrivial blank is used.

In the following examples, the function call, the output from the function (which becomes input to FORMAT, and the output from FORMAT are shown.

Call	SPREAD Output	FORMAT Output
|SPREAD('spread')	s¬p¬r¬e¬a¬d	s p r e a d
|SPREAD(ABC,¬¬)	A¬¬B¬¬C	A B C

STACK

The STACK function allows the values of FORMAT variables to be saved and restored. With it, it is possible to save the value of some variable, such as the tab stops, change the value of the variable, and then later restore the original value, without knowing what that value was.

• Saving a value

```
|STACK(PUSH,var)
```

"var" is the name of the FORMAT variable whose value is to be saved. The list of variables which may be saved is given in the section "FORMAT Variables." TABS and INDENTS are the most commonly saved variables.

• Restoring a value

```
|STACK(POP,var)
```

"var" is the name of the FORMAT variable whose value is to be restored. The most recently saved value of the variable is restored. Values are saved in a stack, so older values may be restored by further calls. For example,

```
TABS 10 15 20
GO
some text |STACK(PUSH,TABS) /V/  saves tab values 10,15,20
TABS 13 18 25
GO
some text |STACK(PUSH,TABS) /V/  saves tab values 13,18,25
TABS 20 25 30
GO
some text |STACK(POP,TABS)       restores tabs to 13,18,25
some more text |STACK(POP,TABS)  restores tabs to 10,15,20
```

April 1977

USER-DEFINED FUNCTIONS

The user may extend the FORMAT language by writing his own functions. Like FORMAT functions, user-defined functions are called from text or control phrase mode, may take parameters, can access FORMAT variables such as the number and tab settings, and may return FORMAT source lines.

Parameters

When a function is called, it is passed a single parameter--the character string parameter list just as it appears in the input. For instance, if

```
|FF(abc,4'a title')
```

appears in the input, the function FF is called with "abc,4,'a title'" as a parameter. The string is passed as a halfword length followed by the character string (just as the PAR=field of a $RUN command). An S-type calling sequence is used. If the function called gives nonzero return code, an error message is printed in the control phrase listing. Any input to FORMAT produced by the function (see below) is processed regardless of the return code. Any value returned by the function (GR0) is ignored.

Generating FORMAT Input

Functions may enter text into the FORMAT input stream by writing the text on SPRINT. When a function is loaded, the normal SPRINT subroutine is replaced by a FORMAT subroutine which inserts lines into the FORMAT input stream.

Accessing FORMAT Variables

Two subroutines are provided for obtaining and changing the values of a number of FORMAT variables. These subroutines have the calling sequences

```
CALL GFMTVR(fmtvar,var,len)
CALL PFMTVR(fmtvar,var,len)
```

GFMTVR obtains the value of a variable, and PFMTVR changes it. "fmtvar" is the name of a FORMAT variable. It is a character string 8 characters long, padded on the right with blanks. The variables which may be accessed are listed in "FORMAT Variables." In GFMTVR, "var" is a variable or array into which the current value of "fmtvar" is to be placed. The optional parameter "len" specifies the length, in bytes, of "var". If "len" is specified, only "len" bytes are returned; otherwise, the entire variable is returned. In PFMTVR, "var" is the new value of "fmtvar". "var" should be the correct type for the variable; no

conversions are made. If the optional parameter "len" is specified, only the first "len" bytes of "var" are used.

The usual S-type calling sequence is used. The high-order byte of the address of the last argument should be X'80' (FORTRAN automatically does this).

Example

The function SPAGE shown below has the calling sequence

```
|SPAGE(side)
```

where "side" is ODD or EVEN. SPAGE skips a page, making sure that the next page number is odd or even as specified by "side".

SPAGE calls GFMTVAR to get the page number. The page number returned is the next one to be used. If it is not as specified by "side", the page number is incremented and the new value set using PFMTVR. Then the function does the page skip, first entering text mode, if necessary. Finally, if SPAGE was called from control phrase mode, it returns to control phrase mode before exiting.

```
      SUBROUTINE SPAGE(WHAT)
C
      INTEGER ODD, PAGE
      INTEGER*2 H2/2/, H3/3/, WHAT(2), EV/'EV'/, OD /'OD'/
      LOGICAL*1 SWV
C
      ODD = 1
      IF(WHAT(2) .EQ. EV) GOTO 11
      ODD = 0
      GOTO 12
11    IF(WHAT(2) .EQ. OD) GOTO 12
      RETURN 1
C
12    CALL GFMTVR('PAGE    ',PAGE)
      IF(ODD .EQ. PAGE-2*(PAGE/2)) GOTO 13
      PAGE = PAGE + 1
      CALL PFMTVR('PAGE    ',PAGE)
C
13    CALL GFMTVR('SW.V     ',SWV)
      IF (SWV) CALL SPRINT('GO',H2,0)
      CALL SPRINT('/S/',H3,0)
      IF (SWV) CALL SPRINT('/V/',H3,0)
      RETURN
      END
```

FORMAT VARIABLES

The following table specifies the names of FORMAT variables that are accessible via the GFMTVR and PFMTVR subroutines in user-defined functions. These are also names that the user would specify to save and restore the value of FORMAT quantities with the STACK function. Unless otherwise specified, each variable or array element is a fullword integer (INTEGER*4).

Name	Use
CARD	Is an array of 2 elements containing the card column limits set by the CARD control phrase.
COLUMN	Is the current column being generated.
COLUMNS	Is the number of columns being printed.
COUNTERS	Is a 6-element array containing the FORMAT counters, |F0 to |F5.
INDENTS	Is an array of 20 elements that contains the indent columns in the form of left-indent, right-indent pairs. Zero means an indent is not set.
LINE#	Is the logical line number of the next line to be printed.
PAGE	The next page number. This is usually more than the current page number.
SPACE	Is the current spacing set by the SPACING control phrase.
SW.AT	Is a byte (LOGICAL*1) switch indicating if FORMAT is currently capitalizing because of a /@/ command operand or /¢/ command operand. It is 0 (.FALSE.), if FORMAT is not capitalizing, and nonzero (.TRUE.), if FORMAT is capitalizing.
SW.M	Is a byte (LOGICAL*1) switch indicating whether FORMAT is currently centering because of a /M/ command operand. It is 0 (.FALSE.), if FORMAT is not centering, and nonzero (.TRUE.), if FORMAT is centering.
SW.PAGE	Is a byte (LOGICAL*1) switch indicating whether the page number is to be printed. It is 0 (.FALSE.), if the page number is not being printed, and nonzero (.TRUE.), if it is being printed.

SW.RIGHT	Is a byte (LOGICAL*1) switch that indicates that the next page is to be a righthand page. It is nonzero (.TRUE.), if the next page is to be a righthand page, and 0 (.FALSE.) otherwise.
SW.U	Is a byte (LOGICAL*1) switch indicating whether FORMAT is currently underscoring because of a /U/ command operand. It is 0 (.FALSE.), if FORMAT is not underscoring, and nonzero (.TRUE.), if FORMAT is underscoring.
SW.V	Is a byte (LOGICAL*1) switch indicating whether FORMAT is in control phrase mode. It is 0 (.FALSE.), if FORMAT is in text mode, and nonzero (.TRUE.), if FORMAT is in control phrase mode.
TABS	Is an array of 20 elements that contains the tab stops. Zero means that a tab stop is not set.
VPAGE	Is the user page number subroutine address. This may be set by a user's page function so that the user may obtain control after a page skip. If there is no user page routine, it is zero.
WIDTH	Is the total column width.
WIDTHS	Is an array of 6 elements containing the individual column widths.

April 1977

CHANGING THE TRANSLATE TABLES

The TRANSLATE control phrase is used to alter three of the translate tables. These tables may be altered to permit easy entry of a frequently used special character, prevent some characters (such as .) from being underscored, produce an overprinted character (such as ⌀), etc. The three tables are called INPUT, OUTPUT, and UNDERSCORE.

All text input is translated using the INPUT translate table. By default, this translation

(1) changes ¬ and X'EO' to X'8A',
(2) changes _ to X'9A', and
(3) changes uppercase letters to lowercase.

The first two changes are for FORMAT's internal purposes. The third is disabled by the LOWERCASE control phrase. Changing the INPUT translate table affects this translation. This translation is the first thing done to an input line, before any other processing by FORMAT. When a line of text is ready to be printed out, it is translated using the OUTPUT table. This translation is done because FORMAT encodes characters for internal purposes and the characters must be returned to normal before printing. For instance, any character which is to be underscored is represented in a special way. The line is also translated using the UNDERSCORE table (the line, not the result of translation with OUTPUT). This translation produces the underscores for an overprinted line. All overprint characters are produced this way, not just underscores; "overprint" table might be a better term. For example, a capital "A" which is to be underscored is represented internally by X'41'. With the OUTPUT table, X'41' is translated into X'C1', or A, and with the UNDERSCORE table it is translated into an underscore. A capital "A" which is not be to underscored is represented internally as itself, X'C1', translated with OUTPUT into itself, and by UNDERSCORE into a blank. An overprint line is actually printed whenever the result of translating with the underscore table contains at least one nonblank character.

The INPUT table is changed by the control phrase

TRANSLATE INPUT loc c

where "loc" is the location in the INPUT table to be changed, and "c" is the new value. "loc" and "c" have either of the formats

(1) a single character enclosed in primes, or
(2) a two-digit hexadecimal number.

Changes made to the INPUT table do not change the characters produced by the |nn special operator, since the INPUT table translation is done before the input line is processed by FORMAT. For example, if

```
TRANSLATE INPUT '#' '%'
```

were given and no other translate tables were changed, then the input line

```
40# of the batch jobs
```

would produce output

```
40% of the batch jobs
```

This translation might be used if a "%" was unavailable on the terminal used for input and to be frequently used in a document but # was not; entering a single character is more convenient than using the special character codes. This control phrase could also have been entered as

```
TRANSLATE INPUT '#' 6C
```

It could be "undone" with

```
TRANSLATE INPUT '#' '#'
```

Whenever the LOWERCASE control phrase is used, the effects of all previous changes to the INPUT table are nullified.

The OUTPUT table is changed just as the INPUT table is. The control phrase is

TRANSLATE OUTPUT loc c

where "loc" and "c" are as before. This translate table is used for lines with FORMAT's internal character encodings, so care must be taken that the right entry is changed (the internal encodings are shown in Appendix E, "Internal Codes"). For instance, if

```
TRANSLATE OUT '%' '#'
```

were given, then a % will be translated into a # on output (whether it was input as a %, produced by a change to the INPUT table, or produced with special characters codes), but a % which is to be underscored will not be translated into a #. To effect that, both the above control phrase and

```
TRANSLATE OUTPUT EC '#'
```

must be given.

It is possible to "work around" a special operator by changing the INPUT and OUTPUT tables. Suppose ¢ frequently appears in the document as a character. Then, by using the control phrases

```
TRANSLATE INPUT '¢' 90
TRANSLATE OUT 90 '¢'
```

a ¢ appearing in the input will appear in the output (if it is not to be underscored), but a ¢ will not be interpreted by FORMAT as a special operator or command operand.

Changing the underscore table is a little more complicated. The format of the control phrase is the same as before:

TRANSLATE UNDERSCORE loc c

"c" is still the new value to be inserted into the table. However, "loc" is not the location in the UNDERSCORE table itself, it is the internal representation of the character. For example, if "a" is to be "underscored" with a "|" instead of "_", the appropriate control phrase is

```
TRANSLATE UND 'a' '|'
```

not

```
TRANSLATE UND 01 '|'
```

If it is desired never to have periods underscored, the appropriate control phrase is

```
TRANSLATE UND 4B ' '
```

while if every period is to be underscored, it is

```
TRANSLATE UNDER 4B '_'
```

These tables may be used together to create characters which require overstriking, such as ⊬ (a combination of b and /). Suppose the character % will not be used in a document. If the following control phrases are given

```
TRANSLATE IN '%' EC
TRANSLATE OUT EC 'b'
TRANSLATE OUT '%' 'b'
TRANSLATE UND '%' '/'
TRANSLATE UND EC '/'
```

then the input

```
where % represents one or more blanks
```

produces the output

```
where ⊬ represents one or more blanks
```

The third and fifth control phrases are included so that a % which gets underscored (with a command operand or special operator) will still appear as ⊬.

APPENDIX A: SUMMARY OF CONTROL PHRASES

Control Phrase	Defaults
BETWEEN g1 g2 g3 g4 g5	gi=2
BOX n b1 b2 ... b20	
CALL name [(parlist)]	
[NO] CAPITALIZE	CAPITALIZE
CARD x y	x=1, y=255
CLEAR	
COLUMNS	c=1
COMMENT text	
[NO] CONTROL	CONTROL
COUNTERS c0,c1,c2,c3,c4,c5	
[NO] CYCLE	NO CYCLE
DEFINE name [n]	
DROP dc	dc='.'=4B
ERASE name	
[NO] ERRORS	NO ERROR
FIGURE fb fe	fb=0, fe=0
FOOTER	
FOOTNOTE fs	fs=0
[NO] FULL	NO FULL
[NO] GENERATE	NO GENERATE
GO	
[NO] HYPHENATION	HYPHENATION
INCLUDE fdname	
INDENTS l1 r1 l2 r2 ... l10 r10	
[NO] JUSTIFICATION	JUSTIFICATION
[NO] KEYPUNCH	NO KEYPUNCH
[NO] LEFT	NO LEFT
LFOOTER	
[NO] LOWERCASE	NO LOWERCASE
LINES l	l=60
[NO] LIST	(batch) LIST; (terminal) NO LIST
LOAD name fdname entry	fdname=ETC:FMTLIB, entry=name
LTITLE tl tc	tl, tc=1
MTS	

[NO] NUMBER	NO NUMBER
[NO] PAGE pn pl pc	pn=1, pl=1, pc=0
PARAGRAPH sp	sp=5
[NO] REPEAT	REPEAT
RFOOTER	
[NO] RIGHT	RIGHT
RTITLE tl tc	tl=1, tc=1
[NO] SAVE	NO SAVE
SENTENCES ss	ss=1
SEPARATION lp	lp=2
SET name = 'text'	
SPACING ls	ls=1
SUBTITLE sl sc	sl=1, sc=1
[NO] SUPER	SUPER
TABS t1 t2 ... t20	
TEXT f p	f=5, p=5
TITLE tl tc	tl=1, tc=1
[NO] TRAIN	NO TRAIN
TRANSLATE table loc c	
[NO] UNDERSCORE	NO UNDERSCORE
VERSION v	v=0
WIDTH w1 w2 w3 w4 w5 w6	w1=64, w2=w3=w4=w5=w6=0
WORD sw	sw=1

APPENDIX B: SUMMARY OF COMMAND OPERANDS

```
   A -- enter "as-is" mode
   B -- backspace
  Bn -- open or close box "n"
C,Cn -- go to next or n-th column
D,Dn -- tab to next or n-th tab stop, leaving the drop character
   E -- end text hold (or FORMAT job)
   F -- begin or stop capitalizing first letter or each word
   G -- justify current line, then go to a new line
H,Hn -- stop indenting, or begin indenting with n-th indent pair on
        next line
I,In -- stop indenting, or begin indenting with  n-th  indent  pair
        immediately
J,Jn -- go to new line once or "n" times
L,Ln -- go  to  new line once or "n" times, unless currently at the
        top of a page
   M -- begin or stop centering text
  O1 -- switch capitalize-next-letter switch
  O4 -- enable capitalize-next-letter switch
  O5 -- disable capitalize-next-letter switch
  O6 -- disable fixed-column switch
   P -- begin new paragraph
   Q -- align right current line
   R -- disable operands F, H, I, M, U, X, @, ¢
   S -- skip to a new page and disable operands F, H, I, M,  U,  X,
        @, ¢
T,Tn -- tab to next or n-th tab stop
   U -- begin or stop underscoring text
   V -- enter control phrase mode
W,Wn -- skip to next page, if less than "n" lines remain on current
        one
   X -- begin or stop suppressing all blanks between words
   @ -- begin or stop capitalizing text
   ¢ -- begin or stop capitalizing text
  |n -- begin  or  stop  printing  out  revision  bars,  if current
|       version <n
  !n -- begin or  stop  printing  out  revision  bars,  if  current
|       version <n
   - -- disable following command operand
```

APPENDIX C: SUMMARY OF SPECIAL OPERATORS

```
        @ -- capitalize next character, if it is a letter, or super-
             script, if it is a number
        ¢ -- capitalize next character if it is a letter, or superscript
             if it is a number
        _ -- underscore preceding character
        ¬ -- nontrivial blank
      |nn -- print special character "nn"
      !nn -- print special character "nn"
    x|@nn -- repeat "x" by "nn" times
|   x!@nn -- repeat "x" by "nn" times
      |Fn -- print and increment counter "n"
       |T -- print time
       |D -- print date
       |- -- marks place where hyphenation may occur
```

APPENDIX D: SPECIAL CHARACTERS

	nn	TN	PN[1]	hex	name
	10	⁽	*	8D	superscript left parenthesis
	11	⁾	*	9D	superscript right parenthesis
	12	⁺	*	8E	superscript plus sign
	13	⁻	*	A0	superscript minus sign
	14	{	*	8B	left brace
	15	}	*	9B	right brace
	16	[	*	AD	left bracket
	17	]	*	BD	right bracket
	18	≤	*	8C	less than or equal to
	19	≥	*	AE	greater than or equal to
	20	±	*	9E	plus or minus
	21	≠	*	BE	not equal to
	22	┗	*	AB	lower-left corner
	23	┛	*	BB	lower-right corner
	24	┏	*	AC	upper-left corner
	25	┓	*	BC	upper-right corner
	26	╋	*	8F	intersection
	27	━	-	BF	horizontal line (not minus)
	28	¤	*	9C	open box
	29	▪	*	9F	filled box (square bullet)
	30	•	*	AF	filled circle (bullet)
	31	°	*	A1	degree
*[3]	32	&	&	50	ampersand
	33	|	|	4F	vertical bar
*	34	¬	¬	5F	logical not
*	35	<	<	4C	less than
*	36	=	=	7E	equals
*	37	>	>	6E	greater than
*	38	+	+	4E	plus
*	39	(	(	4D	left parenthesis
*	40	)	)	5D	right parenthesis
*	41	"	"	7F	double quote
*	42	’	’	7D	single quote
	43	¢	¢	4A	cent sign
*	44	#	#	7B	hash mark
*	45	%	%	6C	percent
	46	@	@	7C	at sign
	47	_	_	6D	underscore
*	48	;	;	5E	semicolon
*	49	:	:	7A	colon
*	50	?	?	6F	question mark
*	51	!	!	5A	exclamation mark
	52	┫	*	2C	right join[2]
	53	┣	*	3C	left join[2]
	54	┻	*	2B	lower join[2]
	55	┳	*	3B	upper join[2]

April 1977

---

[1] When printed with the TRAIN control phrase in effect
[2] Generated by overprinted corners (codes |22,...,|25)
[3] Usually, these characters are entered directly.

APPENDIX E: INTERNAL CODES

The following table gives each of the characters in the FORMAT character set and its internal code in hexadecimal. For characters that are not underscored, this code is the normal EBCDIC representation. For example, the uppercase "A" has the code X'C1' and the lowercase "a" has the code X'81'. In most cases, the underscored characters are coded as the normal EBCDIC code, exclusively or'ed with X'80'. Thus, "a" has the internal code X'01'.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00		a	b	c	d	e	f	g	h	i	"	{	≤	(	+	┼
10		j	k	l	m	n	o	p	q	r		}	¤	)	±	▪
20	-	°	s	t	u	v	w	x	y	z		├	┬	[	≥	•
30	0	1	2	3	4	5	6	7	8	9		┴	┤	]	≠	—
40	∅	A	B	C	D	E	F	G	H	I	¢	.	<	(	+	|
50	&	J	K	L	M	N	O	P	Q	R	!	$	*	)	;	¬
60	-	/	S	T	U	V	W	X	Y	z		,	%	_	>	?
70	0	1	2	3	4	5	6	7	8	9	:	#	@	'	=	"
80	CMD	a	b	c	d	e	f	g	h	i	NTB	{	≤	(	+	┼
90		j	k	l	m	n	o	p	q	r	UNL	}	¤	(	±	▪
A0	-	°	s	t	u	v	w	x	y	z		└	┌	[	≥	•
B0	0	1	2	3	4	5	6	7	8	9		┘	┐	]	≠	—
C0		A	B	C	D	E	F	G	H	I	¢	.	≤	(	+	┴
D0	&	J	K	L	M	N	O	P	Q	R	!	$	*	)	;	¬
E0	-	/	S	T	U	V	W	X	Y	Z		,	%	_	≥	?
F0	0	1	2	3	4	5	6	7	8	9	:	#	@	'	=	EOL

blank = not used
CMD = command operand indicator
NTB = nontrivial blank
UNL = underscore special operator
EOL = end-of-line flag
∅ = blank

April 1977

APPENDIX F: CREDITS

The examples used in this description are taken from the following works:

- Alice's Adventures in Wonderland, Charles Dogdson
- Commentarii de Bello Gallico, C. Julius Caesar
- The Complete Humorous Sketches and Tales of Mark Twain, Samuel Clemens
- A Tale of Two Cities, Charles Dickens
- A Connecticut Yankee in King Arthur's Court, Samuel Clemens
- Three Men in a Boat, Jerome K. Jerome
- The Sign of Four, Sir Arthur Conan Doyle
- Lucky Jim, Kingsley Amis
- Remember the Golden Rule, Brant Parker and Johnny Hart
- The Devil's Dictionary, Ambrose Bierce

April 1977

April 1977

TEXT360

INTRODUCTION

TEXT360 is a text-processing system that can expedite the production of publishable documents.

The TEXT360 user specifies the desired format by embedding two types of instructions in the text flow: "edit codes," which usually affect the format of a relatively small area of text, and "alter codes," which are more general instructions dealing with format specifications of a larger scope, such as page depth, column width, etc.

These codes cause the text processor to "format" the textual material accordingly. The formatting capabilities include hyphenation, line justification, column justification, headings, indentions, and one- or two-column page layout. More complex functions include printing horizontal and vertical lines and placement of tabular text for tables and charts.

The general form of an edit code is -A-, where the minus signs delimit the instruction and A represents the specific code desired. Some of the edit codes may also include numbers, but the general form -A- represents an edit code in discussions of TEXT360.

The general form of an alter code is +ALTER+, where the plus signs delimit the instruction and ALTER represents the specific alter code desired. The alter codes also may include numeric data, but the general form +ALTER+ is used in this description to represent alter codes.

In the general discussion of TEXT360 procedures, the term "controls" is used when no distinction need be made between edit and alter codes.

Documents produced by TEXT360 can be updated with the revision controls. The user can delete, replace, and insert text; copy text from one location to another; and replace a word or phrase by an alternative word or phrase each time the original occurs. It is also possible to create front matter listings such as a table of contents or list of figures; reformat an existing document; and merge information from two documents.

The printed output of TEXT360 can be a fully formatted "camera-ready" document. The master file of the document can be created on any on-line storage medium.

TEXT360 consists of the TEXT360 main processor program and the following peripheral programs: TEXT360 Prescan, TEXT360 Print, and TEXT360 Spelling Dictionary.

The TEXT360 main program performs the following four functions:

- It encodes the text input data and updates the old master file.
- It constructs the text lines, processes the controls, and performs hyphenation and justification.
- It constructs the page formats as specified by the user.
- It creates user-specified lists and performs a spelling check.

The TEXT360 Prescan program analyzes the input to the TEXT360 main program. It is designed to detect errors such as incorrect syntax and invalid parameters.

The TEXT360 Print program prints the formatted document.

The TEXT360 Spelling Dictionary program creates and maintains the dictionary required for the spelling check in TEXT360. When a spelling check is requested, TEXT360 compares each 2- through 24-character word of the document with the dictionary. Any word in the input stream that is not listed in the dictionary is printed with its location in the Spelling Check Messages section of the system-generated information at the end of the document.

The TEXT360 program reads the input, interprets the controls, and writes a file containing the formatted document. This file, the print file, can be printed by the TEXT360 Print program, or any other standard print program. A second file, the master file, is also created. It contains an encoded form of the document and is used for subsequent revisions. After the initial run in which the first master file is created, the user need only specify changes to the document. TEXT360 generates a print file and (optionally) a new master file with each run.

The encoded input to TEXT360 is a combination of text and TEXT360 controls. The control delimiters separate the controls from the text.

Throughout this description, the controls are shown as capital letters; they may be specified using either uppercase or lowercase letters since the plus and minus sign delimiters are sufficient to distinguish control codes.

Unless explicitly stated otherwise, a code must not contain blank characters. However, there are certain alter codes of more than one word which contain blank characters to separate the words.

The parameters that must accompany the controls are shown as "n" or "x", where "n" indicates a numeric character and "x" indicates a numeric, alphabetic, or special character. The number of "n's" shown is

the same as the maximum number of decimal digits allowable. The parameter field can vary in length and, since the format of the field is not dependent upon a specific number of digits, leading zeros can be omitted from the field.

The TEXT360 character set is larger than that of the IBM 029 Card Punch and most terminal keyboards. Those characters that cannot be entered directly from a card punch or terminal keyboard must be specified in another way if they are to become part of the text.

The uppercase and lowercase characters of TEXT360 are represented by the IBM 029 alphabetic characters and capitalization control characters. The capitalization control characters are not explicitly needed if a terminal with both upper- and lowercase characters is used. The remaining characters are expressed by combining a slash with an alphabetic or numeric character. The TEXT360 character set and the corresponding punched-character configuration are listed in "Appendix C: TEXT360 Character Set."

All format layouts shown in this description are governed by the following rules of notation:

- Brackets [] indicate an optional field.
- ␣ indicates a required blank character.
- An ellipsis (...) indicates that the preceding unit can be entered one or more times in succession.
- Words composed of all uppercase letters are keywords to be supplied exactly as shown.
- Words composed of all lowercase letters indicate data to be supplied by the user.
- All punctuation is literal and must be entered as shown.

Both the alter codes and the edit codes are shown in an abbreviated form when the code is discussed in general. These shortened forms should not be assumed to be necessarily valid. For the correct syntax of the alter and edit codes, refer to the appropriate discussions in the sections "Alter Codes" and "Edit Codes."

FORMATTING DOCUMENTS

TEXT360 performs the formatting functions according to the format specifications inserted in the input. The following paragraphs describe the different formatting operations that can be performed by TEXT360.

Because of the flexibility of TEXT360 a wide range of formats can be achieved by the combination of different TEXT360 controls. Once a complete understanding of the coding scheme is attained, a user may be able to obtain the same effect in more than one way.

Text Columns

Text can be printed in one or two columns. The number of columns and the width of the column are controlled by the +SWIDTH+ and +DWIDTH+ alter codes. If the user has no column specification, the program defaults to a one-column text format (+SWIDTH+) with a column width of 65 characters.

A particular column format does not have to remain in effect throughout a document. It may be respecified as currently entered, respecified with a new width, or changed to the opposite column format. This does not cause a new page to be started.

If the program is in the midst of formatting a page when a +DWIDTH+ or +SWIDTH+ code is encountered, it formats that portion being processed as a short page. For example, if text is being printed in a two-column format when a new code is encountered, the program recalculates the page depth, divides the text between the columns and, if specified, performs column justification (making the columns even in depth by the insertion of blank lines).

It then skips two lines and formats the remainder of the page as specified by the new code. This formatting effect is shown in Figure 1.

Throughout this description the term "page width" is used. If the format of a page is one column, the page width is equal to the column width. If the format is two columns, the page width is twice the column width plus three (the minimum number of spaces separating the two columns).

Page Depth

The page depth is the number of lines, including blank lines, that can appear in a text column on one page. Page depth is specified by the +DEPTH+ alter code. In addition, 7 lines are reserved at the top of each page for running head matter and 3 lines are reserved at the bottom of each page for running foot material; these 10 lines are not included in the page depth count.

The depth of a page may vary from 25 to 75 lines. If no depth is specified, the program assumes a depth of 50 lines.

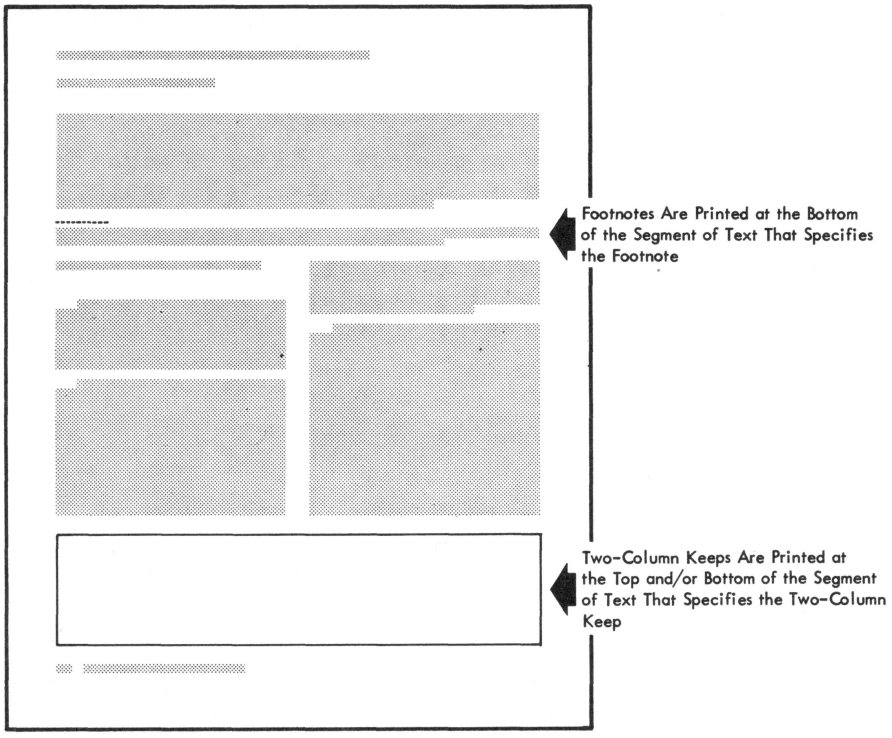


Figure 1. Column Format Change

A page depth of 53 lines is the maximum depth that may be specified for use with a line printer printing 6 lines per inch (the standard setting at the Computing Center). If a page depth greater than 53 is specified, a line printer adjusted to printing 8 lines per inch should be used; if 6 lines per inch printer is used, the text of each page will spill over into a second page.

Only the text column itself is affected by this specification. The program adjusts to the page depth the title, subtitle, date, page numbers, and running feet (such as the chapter title at the bottom of odd-numbered pages).

New Page and New Column

New pages and new columns can be started at any time within a document. The controls that perform these functions are the edit codes -N- and -W-, respectively.

In addition to starting a new page or column, these codes can also force the program to skip pages or columns. Skipped pages are printed with the page number and, when applicable, the running heads (i.e., title and subtitle), date, and running feet.

The text following -N- starts at the beginning of a new page. The text following -W- is placed at the beginning of a new column. If the program encounters -W- while printing text in a one-column format, a new page is started.

Running Heads and Feet

TEXT360 allows for running heads (title, subtitle, date, etc.) and running feet. The running heads are printed at the top of each page. They are aligned with the greatest column width specified on the page.

The +TITLE+ alter code specifies the title to be printed at the top of each text page. The title is printed with six intervening lines between it and the first line of text. It is right-adjusted on odd-numbered (right-hand) pages and left-adjusted on even-numbered (left-hand) pages. Right-adjusted means that the rightmost words are flush with the right margin.

The +SUBTITLE+ alter code specifies the text following it to be printed as a subtitle. The subtitle is aligned with the title, with two blank lines between them. The +DATE+ alter code specifies the text following it to be printed on the same line as the subtitle. The subtitle and date are printed with three blank lines between them and the first line of the text body. The date is left-adjusted on odd-numbered pages and right-adjusted on even-numbered pages.

The +EFOOT+ alter code specifies the running foot to be printed at the bottom of all even-numbered pages. The +OFOOT+ alter code specifies the running foot to be printed at the bottom of all odd-numbered pages. To print a running foot on both even-numbered and odd-numbered pages, an +EFOOT+ code and an +OFOOT+ code have to be specified separately with the same text.

The foot is printed with two blank lines between it and the last line of the page. On odd-numbered pages the foot is printed to the left of the page number, separated by two spaces. On even-numbered pages the foot is printed two spaces to the right of the page number. (The page number, which is generated by the program, is aligned to the greatest column width specified on the page.)

Placement of the title, subtitle, date, and foot is shown in Figure 2. The maximum length of these items is equal to the page width.

It should be noted, however, that the maximum length is computed from page to page. If various page widths are used, the maximum length fluctuates accordingly. Therefore, the maximum page width used within a document should be considered when specifying the length of a running head or foot.

Figure 2. Title, Subtitle, Date, and Foot

Right-Hand Pages

Documents generated by TEXT360 can be formatted so that all pages are right-hand pages. Right-hand pages have the page numbers in the lower right corner, and the titles are right-adjusted. This feature is provided for formatting documents that are to be printed on one side only. The alter code required to specify this feature is +FRONT+.

Margins

A margin is obtained by decreasing the column width. The +MARGIN+ and +LENGTH+ alter codes are used to specify the margins of a document. The margins specified do not have to pertain to the entire document; they can be changed to specify formatted text that needs to be narrower than the width of the column.

The left margin alter code, +MARGIN+, simply instructs the program to print text starting at a certain number of character positions to the right of the leftmost position of the column width.

The +LENGTH+ alter code establishes the right margin by specifying the column position in which the line must end. Both the right margin and the line length are always measured from the leftmost position of the column width.

Printing Text "As Entered"

During normal processing, TEXT360 removes any extra blanks between words in the input. This feature can be suppressed by use of the -A- edit code.

The text following this code is printed "as is." That is, the text controlled by an -A- code is not adjusted with other lines, and extra blanks within the line are not removed. Figure 3 illustrates "as-is" text.

Centered Text

The centering feature automatically centers one line of text between the left and right margins. The edit code that causes centering is -C-.

All leading and trailing blanks are removed from the line of text to be centered; extra blanks within the text are compressed to a single

Encoded Input

```
-P-*THE @GENERIC@@ ATTRIBUTE IS USED TO DEFINE A NAME AS A FAMILY OF ENTRY NAMES
, EACH OF WHICH REFERRED TO BY THE NAME BEING DECLARED. *WHEN THE GENERIC NAME I
S REFERRED TO, THE PROPER ENTRY NAME IS SELECTED, BASED UPON THE ARGUMENTS SPECI
FIED FOR THE GENERIC NAME IN THE PROCEDURE REFERENCE.-AS1-   *GENERAL FORMAT:-AS
1-     @GENERIC@@ (ENTRY/SNAME/SDECLARATION-A-        /V,ENTRY/SNAME/SDECLARATIO
N/B...)-p-...
```

Printed Output

The GENERIC attribute is used to define a name as a family of entry names, each of which is referred to by the name being declared. When the generic name is referred to, the proper entry name is selected, based upon the arguments specified for the generic name in the procedure reference.

General format:

```
GENERIC (entry-name-declaration
   [,entry-name-declaration]...)
```

Figure 3. "As-Is" Text

blank. If more than one line is to be centered, a -C- edit code must precede each line of text.

Text-Line Spacing

Text can be printed single-spaced or double-spaced. The alter codes that control text-line spacing are +SINGLE+ and +DOUBLE+. If no spacing code is specified, the program assumes single-spaced text.

Blank Lines

The blank character holds an important position within the TEXT360 program. The blank character occupies a position within a text line and, although it is not graphically expressed on the printed page, it is considered a printable character. This understanding is important, because the blank character can be used to obtain a blank text line.

Throughout this description, blank lines are referred to as blank text lines or skip lines. A blank line caused by one or more blank characters is a blank text line. A blank line caused by the edit code that causes lines to be skipped is a skip line.

A third type of blank line also exists. This line, which is inserted by the program during formatting functions, is controlled completely by TEXT360. The user controls only the blank text lines and skip lines.

Each type of blank line is assigned a characteristic by the program and is processed in a separate manner. Because the skip line is generally used as a separator between elements of text (e.g., between paragraphs), it is not required if it appears as the first or last line of a column; consequently, it is ignored by the Print program.

For example, if two paragraphs are separated by a one-line skip and the first paragraph ends in the last line of a column, the skip line, which is no longer required, is ignored by the program. If the paragraphs are separated by a blank text line, however, the blank text line appears as the first line of the new column.

Blank lines inserted during formatting are controlled by the program. If blank lines are inserted during a given run and are not required in a subsequent run, they are not regenerated. If a blank line is ignored in one run, it may be inserted in subsequent runs.

Some edit codes start a new line and some do not (see "Edit Codes"). A blank text line can be created by separating two edit codes that start a new line by at least one blank character (no text). For example, the code that causes indentions, -I-, causes a new line to be started. When -I-␣-I- is entered, the first -I- causes the program to begin a new line with the text that follows, in this case a blank character. The program then encounters the second -I- and another new line is started.

An -S- edit code forces the program to skip one or more lines before it prints the text that follows the code. Unless the text is specified with an indention, it is printed at the left margin.

Paragraphs

The format of a paragraph can be specified in several ways. Paragraph format generally consists of a skip of one line and an indention of the first line of text. With the use of the proper edit codes, the user can specify his own paragraph formats.

TEXT360 provides an edit code that combines two formatting functions into one code. The -P- edit code causes the program to skip one line and indent the first line of text following the code three spaces.

Indentions

The TEXT360 program provides two types of indentions. The user can specify a regular indention, which causes only the first line of text following the code to be indented, or a hanging indention, which causes the second and all subsequent lines of the paragraph to be indented (see Figure 4).

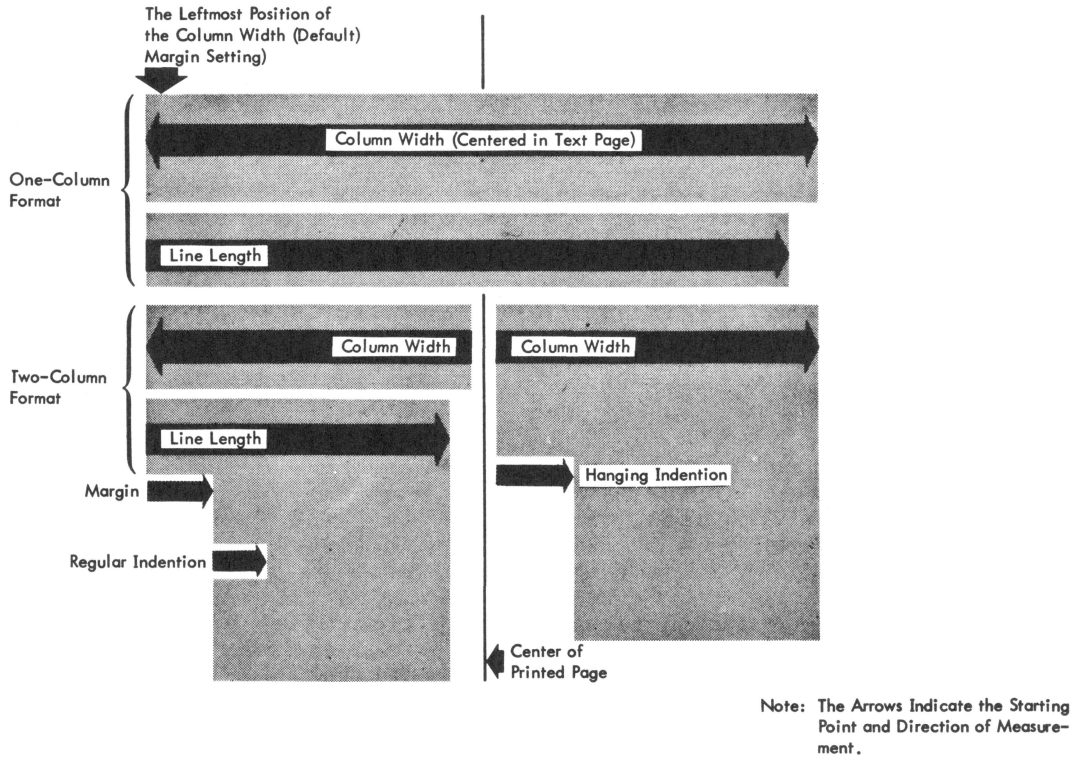


Figure 4. Regular and Hanging Indentions

Both indentions are measured from the current left margin. The regular and hanging indentions are specified by the -I- and -J- edit codes, respectively.

Column Justification

Column justification causes all the text columns on a page to be equal in length by spacing the columns out to the page depth, which is controlled by the +DEPTH+ alter code. The program justifies a column by inserting blank lines within the column. It inserts the additional blank lines where there is a skip line within the column (normally between paragraphs).

The +COLJUST+ alter code invokes the column justification feature. Because the TEXT360 operating mode includes column justification unless otherwise instructed, the +COLJUST+ code is only required to reinstate the feature after it has been suppressed. The +COLJUSTOFF+ alter code suppresses the column justification feature.

Line Justification

Line justification causes all text lines to be spaced out to the line length by the insertion of blank characters between words. The program inserts as many blanks as are necessary to justify the line.

In an attempt to distribute blanks equally across the column, the program inserts an equal number of blanks between each pair of words in the line. If the equally distributed blanks are not enough to cause justification, the program begins at the left end of the line and inserts one additional blank between each pair of words until the line is justified. The program reverses the process in the next line and inserts the blanks from right to left to avoid stacked blank space.

The +JUST+ alter code invokes the line justification feature. Because the TEXT360 operating mode includes line justification unless otherwise instructed, the +JUST+ code is only required to reinstate this feature after it has been suppressed. The +NOJUST+ alter code suppresses the line justification feature.

Occasionally, within a certain portion of a line, the user may not want justification blanks inserted (e.g., the spaces following the numbers of a numbered list). To prevent extra blanks from being inserted, the special blank (entered as "/I") should replace the normal blank. This causes the program to interpret the blank as part of the preceding word, and no extra blanks are inserted.

Tabular Text

TEXT360 provides a set of controls to allow the user to print tabular text. The user specifies the tab settings and indicates the text to be

printed at the settings. The text can either start or end at the tab setting specified (left- or right-justified). The tabular text feature also provides an edit code that allows tabular text and flowing text to be printed in the same line.

The +SETTAB+ alter code specifies the tab settings to TEXT360. The -T- edit code specifies a specific tab position at which the text following it is to be printed. The -.T- edit code specifies the tab position and causes the tabbed-over space to the left of the text to be filled with periods.

The -U- edit code indicates to the program that the tabular text preceding it has been processed and the program is to start formatting the following text in the regular manner, thus allowing two separate pieces of text to be printed in the same line--one, tabular text, and the other, text which flows to the next line below it.

Hyphenation

The hyphenation feature allows the program to hyphenate the last word on a line if the word contains six or more characters (i.e., both halves must contain a minimum of three characters). The program does not hyphenate the last word of a paragraph.

To prevent leaving the second half of a word on a line by itself, TEXT360 does not hyphenate a word immediately prior to an edit code. If the last word of a paragraph must be hyphenated, the user can hyphenate it by specifying the word as two words and insert the hyphen himself.

The program can hyphenate a word incorrectly. Therefore, controls are provided to create local and global dictionaries of the correct hyphenation of incorrectly hyphenated words. The dictionaries are then checked before the hyphenation feature is used.

When line justification and hyphenation are both operative, the program attempts line justification first. However, the program does not insert more than one blank between consecutive words at this point. If the line cannot be justified, the program hyphenates the last word on that line. If the word cannot be hyphenated, it is moved to the next line, and the original line is justified by the insertion of the necessary blanks.

The +HYPHEN+ alter code invokes the hyphenation feature. Because the TEXT360 operating mode includes hyphenation unless otherwise instructed, the +HYPHEN+ code is only required to reinstate this feature after it has been suppressed. The +NOHYPH+ alter code suppresses the hyphenation feature.

The +HYPHALT+ alter code is used to build a local dictionary of correctly hyphenated words that have been incorrectly hyphenated by the

program. The +HYPHCLR+ alter code clears from the local hyphenation dictionary all previously specified words and enters into a new dictionary the properly hyphenated words that follow the code. Whereas +HYPHALT+ can add words to an existing dictionary, +HYPHCLR+ starts a new dictionary of words each time it is specified.

A global hyphenation dictionary also may be used to specify the correct hyphenation of words incorrectly hyphenated by the program. This dictionary, which is contained in an external file, is specified by the HYPHDICT parameter (see "Appendix B: Peripheral Programs").

When a word is to be hyphenated by TEXT360, the program first checks the local dictionary for its correct hyphenation; if it is found, that hyphenation is used. If it is not found, the program then checks the global dictionary; if it is found, that hyphenation is used. If it is not found, the word is hyphenated by the program.

The local hyphenation dictionary is searched in reverse order of word entry, that is, the latest entered word is checked first. The global hyphenation dictionary is searched in the order of word entry, that is, the earliest entered words are checked first.

Words in either the local or global hyphenation dictionary may be from 2 to 24 characters in length. The program itself may hyphenate words that are longer than 24 characters.

Footnotes

TEXT360 allows the user to specify text that is to be printed as a footnote. The program prints the footnote at the bottom of the text column containing the footnote reference (see Figure 5). The program automatically skips one line, prints a line of dashes (cutoff line), then prints the footnote. Blank lines that are to appear between the cutoff line and the first line of the footnote must be specified by the user (with the -S- edit code).

The -F- edit code indicates the beginning of the footnote text. The footnote text should be entered in the input flow at the point of the footnote reference. The -Z- edit code ends the text designated as a footnote.

If two footnotes are over 20 lines, the program ends the page with the first footnote and places the second footnote on the next page.

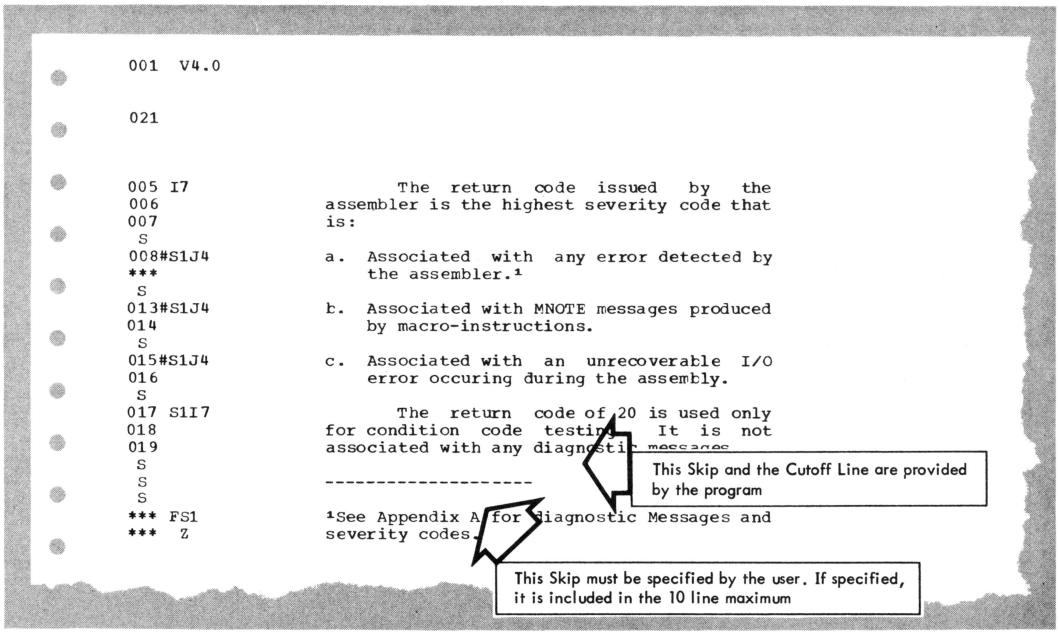


Figure 5. Printed Footnote

HEADINGS

A heading is a line (or lines) set apart from the text body, describing the content of the section that follows it. The format of the heading indicates its relationship to the other headings used in the document. TEXT360 provides two heading sets, each with six heading formats.

Heading Sets

The +HEAD+ alter code is used to select one of the two heading sets. The heading sets are identified by the alphabetic characters "A" and "B" (see Figure 6). Each heading set provides six heading formats numbered from 1 to 6. The difference between the two heading sets is the spacing before and after headings 2, 3, and 4.

Headings 1 to 4 in both sets are "stand-alone" headings. That is, the text that follows them begins on a separate line. Upon encountering a heading 1 to 4, the program formats the text from that point to the next control as the heading. Headings 1 to 4 are not justified or hyphenated.

Headings 5 and 6 are "run-in" headings. The text that follows them continues on the same line. The format of headings 5 and 6 continues until a colon or period followed by one blank is encountered.

Heading Set	Heading Level	Stand-alone	All Cap	Underscore	Skips Before	Skips After	New Page
A	H1	x	x	x	0	5	x
	H2	x	x	x	2	1	
	H3	x	x		2	1	
	H4	x		x	1	1	
	H5		x	x	1	0	
	H6			x	1	0	
B	H1	x	x	x	0	5	x
	H2	x	x	x	3	2	
	H3	x	x		3	2	
	H4	x		x	3	2	
	H5		x	x	1	0	
	H6			x	1	0	

Figure 6. TEXT360 Heading Sets

Heading Formats

The -H- edit code specifies a heading format within the selected heading set. When entering the text that is to be used as a heading, the user should consider the following:

- Headings 1, 2, 3, and 5 are automatically printed uppercase (all capital letters) regardless of how they appear in the input; headings 4 and 6 are printed as specified in the input.

- The program can be instructed to create a table of contents consisting of the headings used in the document (see "Supplemental Listings"). When this feature is used, the headings must be specified exactly as they are to appear in the table of contents, e.g., if the heading is entered in lowercase letters, it will appear in the table of contents in lowercase letters regardless of its appearance in the body of the text.

April 1977

KEEPS

The "keep" feature permits the user to ensure that a specified portion of text, such as a table, is not split between columns or pages. There are three types of keeps. Two types are restricted by the column width, whether in a one-column or a two-column format. The third type allows the insertion of data that is wider than one column in a two-column format.

Regular Keeps

The -K- edit code indicates the beginning of a regular keep; that is, it begins a section of text that the user does not want to be divided between two columns or pages and which should not be moved from its original location within the input text.

When the program encounters a -K- edit code, it examines the remaining portion of the column being processed to determine if the keep will fit. If the remainder of the column is long enough, the keep is inserted. If the remainder is not long enough, the keep is printed at the top of the next column, and the remainder of the previous column is left empty.

Floating Keeps

The -Y- edit code indicates the beginning of a floating keep; that is, it begins a section of text that should not be divided between two columns or pages, but may be moved from its original position within the input text. The program may move the floating keep as much as one column from its original input location. This means it may move across page boundaries.

When a floating keep is placed at the beginning of the next column, the program moves text following the keep to fill the unused portion of the preceding column.

The movement of the floating keep is toward the end of the document. Although the floating keep can move with respect to the text following it, it is assigned page and line numbers for updating purposes at its original location within the input stream.

No blank lines are inserted automatically after the keep. Therefore, the user should provide for the spacing at the bottom of the keep within the context of the keep, e.g., ending the keep with the edit-code combination -S1R-, to space the keep away from the text that follows.

Two-Column Keeps

The -D- edit code indicates the beginning of a two-column keep. A two-column keep is used to insert data wider than the column width when in a two-column format. When a -D- edit code is encountered, the program temporarily modifies the column width to equal twice the specified column width plus three.

The manipulation of the column width is an internal function; it does not affect the specified column width. The program places the keep at either the top or the bottom of the printed page, or at the top or bottom of the segment of text that specifies the two-column keep if there is a change of column format in the middle of the page.

When a -D- edit code is encountered as the left column is being formatted, the keep is printed at the top of the page or two-column text segment, if there is no level 1 heading on that page. If there is, or if the keep is encountered as the right column is being formatted, the keep is printed at the bottom of the page or two-column text segment.

If there is not enough room at the bottom of the page, the keep is printed at the top of the next page, and the remainder of the page is left empty. The keep is separated from the following or preceding text by two blank lines.

In many instances the user may wish to flow text across the full width of a two-column keep. This can be done merely by temporarily increasing the line length. The modified column width allows the line length to be increased to the modified width.

For example, if the line length and column width are set at 43 when the -D- code is encountered, the line length can be increased to 89 (2 x DWIDTH + 3) by specifying the +LENGTH+ alter code. This allows the text to flow across the full width of the two-column keep. When the keep is ended, the line length and margin are automatically reset to the original values, if they were changed within the keep.

If the line length is not increased, text can be made to extend across the full width of the two-column keep by using the -A- code to specify "as-is" text.

A two-column keep encountered while processing in a single-column format is treated as a regular keep. No skip lines are created, and the keep is not moved to the top or bottom of the page in progress. Figure 7 illustrates the placement of regular, floating, and two-column keeps.

Ending Keeps

The -R- edit code releases the regular, floating, and two-column keeps. To prevent a keep from being specified within another keep, the program generates a release for any keep in effect when a second keep code is encountered. However, for best results all keeps should be ended with an -R- code.

If a keep exceeds the page depth, the program attempts to release the keep, and it treats the remainder of the keep as regular text.

TABLES

Tables and charts are usually combinations of tabular text, horizontal lines, and vertical lines. TEXT360 provides a method of drawing table lines by combining lines (|, —), corners (└, ┌, ┘, ┐), and intersections (┼); the program provides the proper intersection at the junction of a horizontal and a vertical line.

Line-drawing codes do not cause a new text line to be started. When a new text line is also required, it must be specified by the user.

Care must also be taken to ensure that a text character does not occupy the same position as a line character. When a conflict exists between text characters and line characters, the text characters override the line characters. The line characters will, however, override the blank characters of a text line, but will not override special blanks (specified as /I).

For example, if the text line "text will not be included." is to be followed by a horizontal line below it, but the user fails to request a new line, the program will print the line as follows:

text—will—not—be—included.———

The table-formatting codes specify column positions, which are measured from the first position in the column width.

Beginning Tables

The -B- edit code begins a table, and the -E- edit code ends a table. A -B- code causes a horizontal line to be printed starting at the leftmost column position specified within the code and ending at the rightmost column position specified. This code also starts a vertical line in each of the column positions specified, which need not be entered in ascending order. Each text line includes the vertical lines until the user stops them.

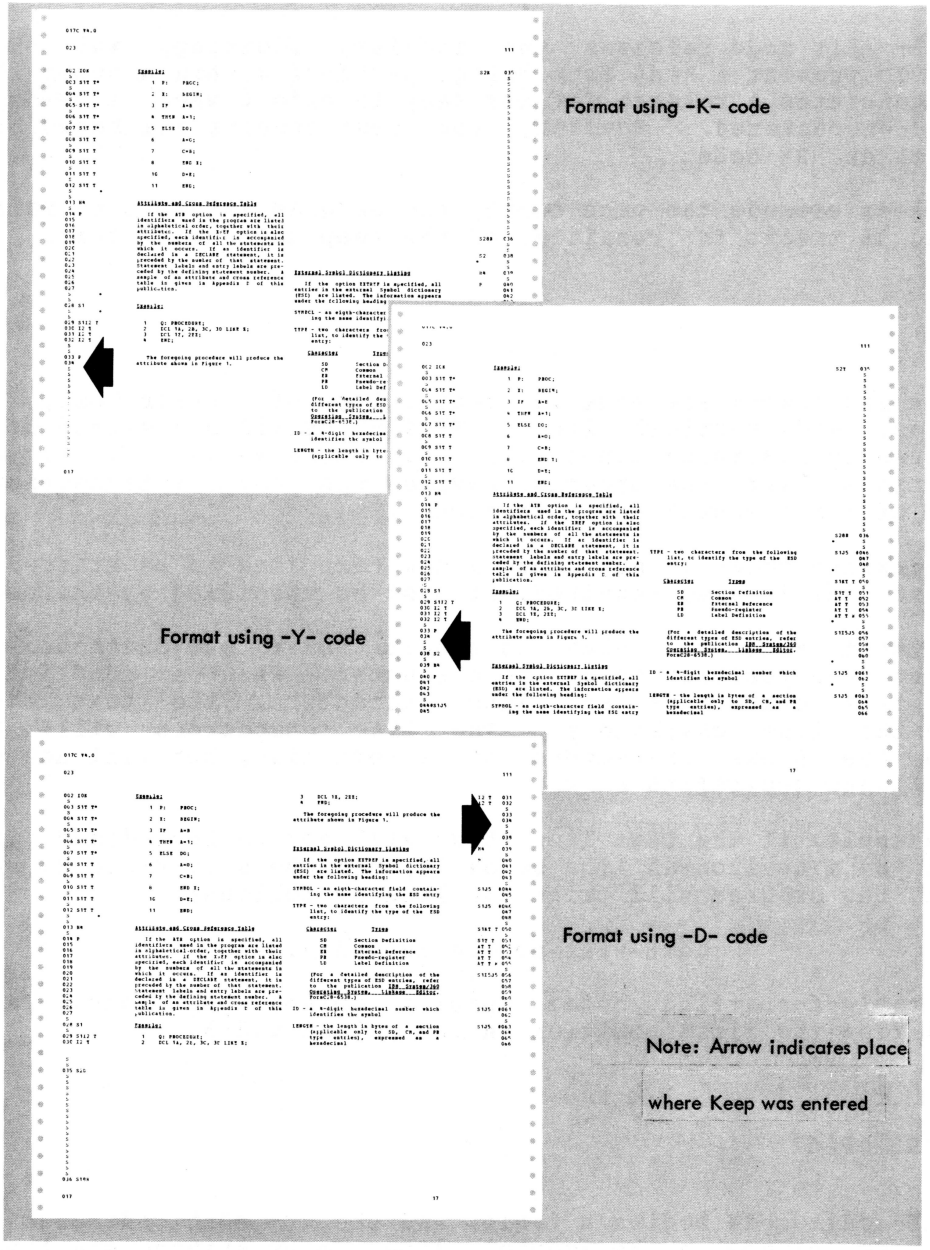


Figure 7. Regular, Floating, and Two-Column Keeps

A subsequent -B- code can be specified while a previous -B- code is still operative. The second -B- code ends all vertical lines of the first before starting the new table. In this fashion, a table of varying format can be constructed by using the -B- code. If the second table is not overlapped by the first, the vertical lines that lie beyond the width of the new table are joined by the horizontal line generated by the second -B- code.

This is shown in Figure 8, where the first -B- code is followed by an -S- code and another -B- code with different specifications. This causes the first table to begin, a skip of three lines, and the second table to begin on the fourth line (thus ending the first table). After another skip of three lines, the second table is ended by the -E- code. This table is generated by the code sequence

```
-S1B1,10,62,72-⊬
-S3B5,67-⊬
-S3E-
```

Figure 8. Table Example 1

The use of the ⊬ in the above and following examples is to prevent an edit-code line from being used in combination with the next edit-code line. See the section "Edit Codes" for a discussion of using edit codes in combination.

A -B- code, combined with a -Bnnn,...,nnn- code to form -BBnnn,..., nnn, causes the program to combine the column positions of the -BBnnn,...,nnn code with those of the last -Bnnn,...,nnn code.

Figure 9 shows what appears to be one table, but in reality is two tables in combination. A -B- code begins the first table. After a skip of three lines, the first table is combined with a second table that begins on the fourth line. After another skip of three lines, the combined table is ended by the -E- code. This table is generated by the code sequence

```
-S1B1,10,62,72-⊬
-S3BB5,67-⊬
-S3E-
```

Figure 9. Table Example 2

Tables can be ended in two ways. As shown in the previous examples, a table can be ended by beginning a second table. Also, the -E- edit code causes all vertical lines started by the -B- edit code to be ended and a horizontal line to be printed between the leftmost and rightmost vertical lines of the table.

Horizontal Lines

Horizontal lines can be printed in four ways. A horizontal line is printed when a table is begun with a -B- code and when a table is ended with an -E- code.

The -L- edit code prints a horizontal line between the leftmost and the rightmost column positions of the most recent -B- edit code. As in all tables, the intersections at the vertical lines are printed by the program. Figure 10 shows a table with a horizontal line drawn across the middle. This table is generated by the code sequence

```
-S1B25,45-ϕ
-S2L-ϕ
-S2E-
```

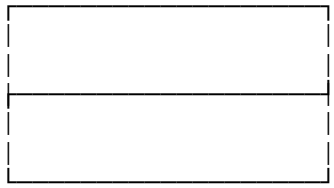


Figure 10. Table Example 3

The -G- edit code causes a series of horizontal line segments to be printed between the column position numbers specified in the code.

Figure 11 shows two lines of horizontal line segments. These are generated by the code sequence

```
-S1G25,45-ǿ
-S1G25,30,35,45-
```

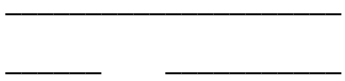


Figure 11. Horizontal Line Example

Vertical Lines

In addition to the vertical lines begun by the -B- edit code, vertical lines can be generated with a -V- edit code. This code prints a vertical line in each column position specified within the code.

If a second -V- code is specified with different column positions while a -V- code is still operative, the first series of vertical lines are simply ended at that point and no horizontal line is drawn. As with the vertical lines printed by the -B- code, each text line will include the vertical lines until the user stops them.

Vertical lines begun by a -V- code are ended either by a second -V- code or by a -Q- edit code. If column positions are specified within the -Q- code, only the vertical lines in those positions are ended. A -Q- code without column positions ends all the vertical lines begun by the most recent -V- code. The vertical lines end in the text line in which the code is specified.

Figure 12 shows a tic-tac-toe diagram draw by using vertical and horizontal lines. This diagram is generated by the code sequence

```
-S1V30,40-ǿ
-S2G24,46-ǿ
-S2G24,46-ǿ
-S2Q-
```

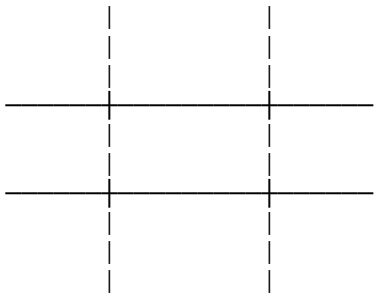


Figure 12. Vertical Line Example

The -Q- code does not end vertical lines started by the -B- edit code. The vertical lines started by the -B- code are ended by the -E- code or another -B- code.

TEXT360 has the restriction that only one table may be be started by the -B- edit code. If more than one box is to be started on a line, the second and subsequent boxes must be drawn using the -G- and -V- edit codes. Figure 13 shows how two boxes may be started on a single line. This figure is generated by the code sequence

```
-S1B20,25G40,45V40,45-ϕ
-S2EG40,45Q-
```

Figure 13: Box, Vertical Line, and Horizontal Line Example

Figure 14 shows a table using box codes, horizontal lines, and vertical lines. A box is drawn around the entire table. Three horizontal lines and a vertical line are used to separate the parts of the table. The table is drawn by the code sequence

```
+SETTAB1=21,2=22,3=32,4=34,5=42,6=51,7=56R+
-S1B20,58-ϕ
-S0T3-@SAMPLE TABLE
-S0LV32-ϕ
-S0T1-QUANTITY-T4-ITEM-T6-PRICE@@
-S0L-ϕ
-S0T2-1 *DOZ.-T4-*EGGS-.T6-/D-T7-.69
-S0T2-2 *LB.-T4-*HAMBURGERS-.T6-/D-T7-1.18
-S0T2-10 *LB.-T4-*POTATOES-.T6-/D-T7-1.69
-S0T2-1 *LOAF-T4-*BREAD-.T6-/D-T7-.25
-S0T2-1 *LB.-T4-*BUTTER-.T6-/D-T7-.70
```

```
-S0T2-4-T4-*APPLES-.T6-/D-T7-.43
-S0LQ-⅁
-S0T5-@TOTAL@@-T6-/D-T7-4.94
-S0E-
```

SAMPLE TABLE		
QUANTITY	ITEM	PRICE
1 Doz.	Eggs.....	$ .69
2 Lb.	Hamburgers.....	$ 1.18
10 Lb.	Potatoes.....	$ 1.69
1 Loaf	Bread.....	$ .25
1 Lb.	Butter.....	$ .70
4	Apples.....	$ .43
	TOTAL	$ 4.94

Figure 14: Tabular Example

Figure 15 shows a more complex diagram using box codes, horizontal lines, and vertical lines. This illustration was taken from the section "Virtual Memory Management" in MTS Volume 5. This diagram is generated by the following code sequence:

```
+SETTAB1=3,2=16,3=30,4=49,5=63,6=19,7=32,8=52,9=65+
+SETTAB10=12,11=26,12=39,13=59+
-S1A-   @MAIN@@
-S0B1,10-ø
-S1T1-@LINK A@@-G10,11T10-1-G13,15V15-
-S0QG3,8,10,11T10-//U
-S0G3,8T10-8-V15-ø
-S0T1-@STOP@@-V12,15-ø
-S0T6-*A-T7-*B
-S0EG14,24,28,37V12,14,24,28,37-ø
-S1T2-@SAVE@@-T3-@SAVE@@
-S0G16,21T3-@LINK C@@-G37,38T12-3-G40,48V12,14,24,28,37,48-ø
-S0G16,21,30,35,37,38T12-5-G40,45V12,14,24,28,37,45,48-ø
-S0T2-@XCTL B@@-G24,25T11-2-G27,28T3-@LINK E@@-G37,38T12-6
-G40,42V12,14,24,28,37,42,45,48-ø
-S0QG30,35,37,38T12-//U-V12,14,24,28,37,42,45,48-ø
-S0G14,24T3-@RETURN@@-T12-7-V12,28,37,42,45,48T8-*C
-S0V12,28,37,39,42,45,48-ø
-S0G28,37,47,57V12,32,39,42,45,47,57-ø
-S0G12,32V39,42,45,47,57-ø
-S0T4-@SAVE@@-T9-*D
-S0G49,55,61,70V39,42,45,47,57,61,70-ø
-S0G49,55-ø
-S0T4-@XCTL D@@-G57,58T13-4-G60,61T5-@SAVE@@
-S0G63,68-ø
-S0G47,57,63,68V39,42,45,61,70-ø
-S0G45,61V39,42,61,70T5-@RETURN@@
-S0T8-*E
-S0G47,57,61,70V39,42,47,57-ø
-S1G42,47V39,47,57T4-@SAVE@@
-S0G49,55-ø
-S0G49,55-ø
-S0G39,47V47,57T4-@RETURN@@
-S1G47,57Q47,57-ø
```

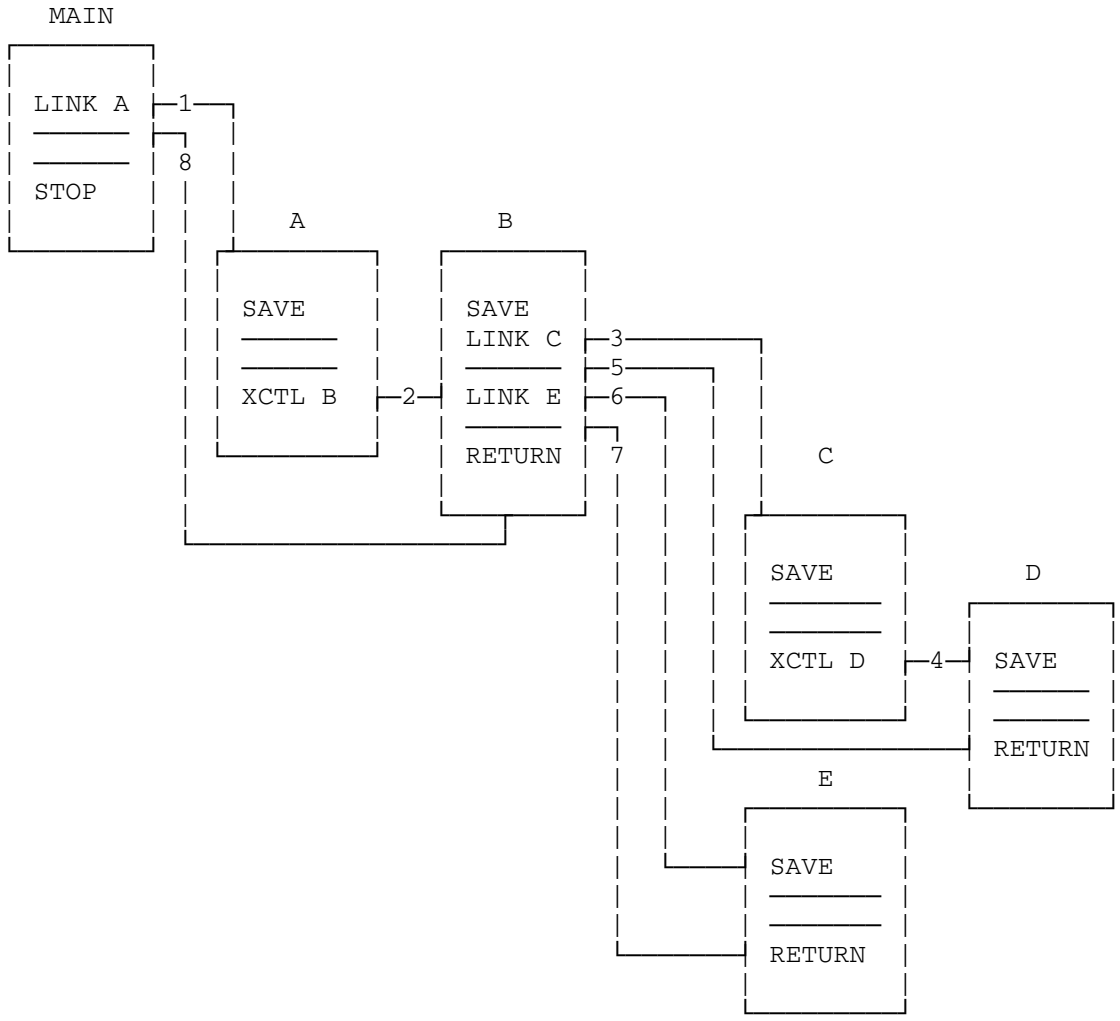


Figure 15: Diagram Example

MISCELLANEOUS FUNCTIONS

TEXT360 provides controls which allow the user to perform operations other than simple formatting. The following paragraphs explain these miscellaneous controls.

User-Defined Controls

Several of a user's formats may require the same combination of one or more edit codes, and these formats may be used repeatedly throughout a document. To eliminate the necessity of repeating the same, sometimes lengthy, combination code at each applicable location, the program provides a method of defining a combination code as one edit code. Once defined, the new code can be entered in place of the lengthy combination code.

The +DEFINE+ alter code allows this definition of a multioperation format function. For example, if -S2I3J8- is a frequently used code, it can be defined by specifying +DEFINE?1=S2I3J8+. Then whenever the user specifies the edit code -?1- in the text, the program interprets the code as -S2I3J8-.

Margin Pointers

TEXT360 provides a margin pointer to assist the user in locating information he wants to find quickly. The pointer is caused by the -M- edit code.

Whenever the -M- code is encountered, the program prints a special blank in the text line in place of the -M- code and a pointer (< or >) next to the line containing the special blank. The pointer is printed in the gutter (the white space separating the two columns) when formatting text in two columns and in the right margin when formatting text in one column.

No blanks are inserted by the program for justification purposes at this point, but the sentence can be "broken" at the end of a line of text, in which case the blank remains, either at the end of the line or at the start of the next line.

Spelling Check

TEXT360 provides a facility for checking the spelling of words in the input stream. The words are compared with a dictionary of words provided by the TEXT360 Spelling Dictionary program (see "Appendix B: Peripheral Programs"). Any word in the input stream not listed in the dictionary (whether correctly spelled or not) is printed with its location in the spelling check messages section of the system-generated information at the end of the document.

If a spelling check is requested, the program loads the dictionary into virtual memory and performs the comparison. The comparison is

invoked by the +SPELL+ alter code. The new master file is rewound and the spelling check is performed by a comparison of the new master file with the spelling dictionary.

The +SPELL+ code causes a significant increase in processing time; therefore, it should be used with some discretion. The more extensive the spelling dictionary is, the higher the processing costs will be.

INPUT TO TEXT360

Input to TEXT360 need not conform to any column restrictions. All column positions of an input line or punched card, or any portion thereof, may be used. The program processes the input continuously without regard to line boundaries (unless the +ASIS+ alter code is in effect).

Thus, for a punched card, column 80 and column 1 of the next card, in effect, are adjacent, and a word or code can be split between cards. When a word ends in the last column of an input line, column 1 of the next line must be left blank. At least one blank is required between words; if more than one blank is used, the program normally removes all extra blanks. (However, the program can be instructed not to remove the extra blanks from the input text; see "Printing Text 'As Entered.'")

The +BLANK+ alter code may be used to cause all input lines to be padded with a single blank. When this edit code is in effect, control codes and text words cannot be broken across input line boundaries.

An exception to the one-blank rule is the procedure for ensuring that there are two blanks at the end of every sentence. The program interprets a period, exclamation point, question mark, or colon followed by one or more blanks as the end of a sentence. When any of the above characters is used in association with a double quotation mark, single quotation mark, or right parenthesis, and is followed by one or more blanks, the end of a sentence is assumed.

Special Control Characters

Six characters are reserved to control the input. These six special control characters are the plus sign (+), minus sign (-), asterisk (*), commercial "at" sign (@), dollar sign ($), and slash (/). Combined with the remaining characters, they are used to express the complete TEXT360 character set, control capitalization and underscoring of words, and delimit the edit codes and alter codes used by the program.

A seventh reserved character (_) is used to control backspacing if the +BACKSPACE+ alter code is in effect.

The special control characters are entered in the input stream directly from a keypunch or terminal keyboard. When one of these special control characters must appear as text, however, it must be specified in its encoded form. For example, the hyphen in "direct-access" must be entered as "/S".

The complete TEXT360 character set and the corresponding keypunch and terminal character configurations are listed in "Appendix C: TEXT360 Character Set."

End Input Delimiter

The end input delimiter may be used to terminate an input line. This is done by entering the special character slash followed by a blank. When the program encounters "/ø", it ignores the remainder of the current line and begins processing at column 1 of the next line.

An input line need not be ended at a logical location (i.e., a word boundary). It can end within a word or even within a control. However, an input line cannot be ended within a character that is expressed by two or more characters. For example, the character slash is expressed by "/Z". This character cannot be entered with the slash, followed by the delimiter /ø, in one line and the Z in column 1 of the next line. However, it can be entered with the slash in that last column of one line and the Z in column 1 of the next line.

The end input delimiter is generally useful when the input is being entered via punched cards, or when it is imperative that the input line not contain a trailing blank. The end input delimiter is also useful for putting comments into the input stream (all text in the input line following the ø is ignored by TEXT360).

Capitalization

Capital letters are controlled by two special control characters: the asterisk (*) and the at sign (@). The asterisk causes the letter immediately following it to be capitalized and is commonly used when one, two, or three letters are to be capitalized. When several consecutive letters are to be capitalized, the at sign is commonly used. The program starts capitalization when a single "@" is encountered and ends capitalization when "@@" is encountered. For example, the capitalization controls can be used to capitalize "word" by specifying either @WORD@@ or *W*O*R*D.

The controls for capitalization are designed to precede and follow text. During a revision, they cannot be inserted or deleted independently of text; rather, the complete text must be manipulated. For

example, to capitalize a phrase in an existing document, the complete phrase must be replaced by the capitalized phrase; the capitalization controls cannot be inserted alone as a revision (see "Document Line Updating").

Capitalization controls have no effect on any characters other than alphabetic characters. For example, "@IBM 29 C@@ARD *PUNCH" is printed as "IBM 29 Card Punch".

Capitalization begun by "@" is ended either by the user, with "@@", or by the program, when it encounters the next asterisk or the next alter code.

Restriction:

- The asterisk should immediately precede an alphabetic character, the underscoring special control character "$", or the special character that indicates end of input "/⊬"; if the asterisk precedes any other character, including a blank character, the asterisk is ignored.

The +CONVERTOFF+ alter code may be used to specify that the text is being entered directly in upper- and lowercase (usually from a terminal with both an upper- and lowercase character input feature). When +CONVERTOFF+ is in effect, the asterisk or at sign special control characters are not needed to indicate capitalization. The +CONVERT+ alter code may be used to resume normal (default) processing. When +CONVERT+ is in effect, all input text is converted to lowercase before it is processed by TEXT360.

Underscoring

Underscoring is invoked by the special control character dollar sign "$". The program starts underscoring when a single "$" is encountered and ends it when "$$" is encountered, or when it encounters either an alter code or an edit code. When underscoring and capitalization controls are used together, the order of specification is interchangeable (e.g., $@ and @$ are equivalent).

The controls for underscoring are designed to precede and follow text. During a revision, they cannot be inserted or deleted independently of text; rather, the complete text must be manipulated. For example, to underscore a phrase in an existing document, the complete phrase must be replaced by the underscored phrase; the underscoring controls cannot be inserted alone as a revision (see "Document Line Updating").

Backspacing

Backspacing (overprinting) is controlled by the +BACKSPACE+ alter code. When +BACKSPACE+ is in effect, backspacing is invoked by the special control character underscore (_). In this manner, it is possible to overprint one character with another character, e.g., *O_/Z is printed as "Ø". When +BACKSPACE+ is in effect, the literal underscore in the input text stream must be entered as "/J". The +BACKSPACEOFF+ alter code may be used to disable the backspacing feature.

Entering Direct Text

The direct (untranslated) text entry feature is controlled by the +ASIS+ alter code. When +ASIS+ is in effect, all input text lines are treated as literal input lines with each input text line starting a new line in the document. All special characters are treated as literal characters; hence, the capitalization, underscoring, and backspacing functions are inoperative. Upper- and lowercase text is entered directly as specified without any case conversion being performed. Direct text entry may be disabled by the +ASISOFF+ alter code. The +ASISOFF+ alter code is the only control code recognized when the +ASIS+ alter code is in effect.

Input Text Suppression

The input text suppression feature is controlled by the +IGNORE+ alter code. When +IGNORE+ is in effect, all input from the input source stream is ignored until the +IGNOREOFF+ alter code is encountered. The ignored text is not entered into the document and is not retained in the master file. The +IGNOREOFF+ alter code is the only control code recognized when the +IGNORE+ alter code is in effect.

The input text suppression feature is useful for suppressing selected portions of a document or for inserting comments into the input text stream.

OUTPUT OF TEXT360

The output of TEXT360 is a master file and a listing of the print file of the document. In addition to the listing of the document text, there are supplemental listings that TEXT360 creates according to user instructions and system-generated information and error messages.

April 1977

Logical Sheets

TEXT360 treats a document as a series of logical sheets. That is, each odd-numbered page and its even-numbered backup page (e.g., 15 and 16) are treated as an individually formatted unit that is independent of the format of the next unit. The subsequent deletion of text from a logical sheet can cause a blank or partial backup page; the addition of text can cause supplementary backup pages to be generated.

The logical sheet mode is used principally during revision activity. It allows any odd-numbered page and its even-numbered backup page within the document to undergo extensive revision without affecting the remainder of the document. In this way, only the changed pages need to be printed to produce a revised document.

When a logical sheet overflows, due to the addition of text, the program generates supplementary pages. The supplementary pages are numbered with the even-page number of the logical sheet and a suffix of .1, .2, etc.

For example, if text is added to page 15 and the additional text causes page 16 to overflow, the program generates as many supplementary pages as are necessary to hold the overflow. The supplementary pages are numbered 16.1, 16.2, 16.3, etc.

A page plus all of its supplementary pages can hold a maximum of 999 lines. If this is exceeded, the program will automatically switch out of logical sheet mode.

If the last supplementary page has an odd suffix (e.g., 16.3), the program generates a blank, even-numbered backup page (i.e., 16.4). This is to prevent subtitles and running feet from getting out of step with odd and even pages. In other words, if the blank page were not produced, when the resulting document is printed on both sides of the paper, an odd-numbered page could appear as a left-hand page.

If text is added to page 15 and the overflow to page 16 does not cause page 16 to overflow, the logical sheet does not overflow and no supplementary pages are generated.

Overriding Logical Sheet Mode

It should be clearly understood that a logical sheet is not necessarily a "physical" odd and its even backup page. For an initial run, if nothing is done to force the program out of the logical sheet mode, the program attempts to treat the entire document as one logical sheet (i.e., two pages). For this reason, the +PAGE+ alter code, which forces the program out of the logical sheet mode, should precede all input text for an initial run.

The +PAGE+ code forces the program out of the logical sheet mode from the point where it is encountered to the end of the document. Out of the logical sheet mode, the program flows text from physical page to page without leaving blank pages or generating supplementary pages. The physical pages (i.e., odd and even backup pages) are then treated as logical sheets on successive runs.

The +PAGE+ code is not retained on the master file. It affects only the run in which it is specified. Unless the +PAGE+ code is specified, the program operates in the logical sheet mode.

The +REPAGE+ alter code is used to force the program out of the logical sheet mode until it reaches the page number specified within the code. When the program encounters this code, it flows text from page to page without leaving blank pages or generating supplementary pages.

The +REPAGE+ code is generally used to update an existing document, because it is more economical than the logical sheet mode in the number of supplementary pages generated. For example, if a chapter of a publication is updated in the logical sheet mode, three or four extra sheets might result with blank backup pages. However, if the entire chapter is run in the +REPAGE+ mode, there may be only one extra sheet generated; or, if the changes are small, there may be none.

Because a logical sheet consists of an odd-numbered page and its even-numbered backup page, the +PAGE+ code should specify an odd-numbered page (since it instructs the program at what page to start "repaging"), and the +REPAGE+ code should specify an even-numbered page (since it instructs the program at what page to stop repaging). However, if no number is specified with the +PAGE+ alter code, the program automatically starts page numbering with the number of the page in progress.

The +REPAGE+ code is not retained on the master file. It affects only the run in which it is specified.

Revision Bars

Revision bars (|) are assigned by the program to identify lines that have been changed. A revision bar is not assigned to the lines that have been deleted, copied (without change), or rearranged as the result of format changes (e.g., new line length).

Revision bars are accumulated from one run to the next. Because they are accumulated and do not distinguish current changes from previous changes, the program also prints a plus sign (+) in the control field to identify lines changed during the current run (see "Control Fields").

The accumulated revision bars can be eliminated by the +NEWLEVEL+ alter code. The +NEWLEVEL+ code causes all previously accumulated

revision bars to be cleared. However, revision bars are printed for the changes entered in the run in which +NEWLEVEL+ is specified.

Supplemental Listings

User-specified supplemental listings can be created by TEXT360. Tables of contents and lists of figures are generally produced by means of this feature.

The user first identifies the text that is to be printed in a supplemental listing when the text is entered in the input stream, then instructs the program to print the listing.

Specifying the Text:

The -X- edit code is used to identify lines that are to appear in a supplemental listing. The user includes a number (1 to 9) or alphabetic character in the -X- code for identification purposes. Supplemental listings are requested by group, identified by the number or alphabetic character.

In addition to the lines the user tags with the -X- code, the program automatically tags all headings. In other words, there is an embedded -X- code of equal value with every -H- code, so that headings can be listed to form a table of contents.

Printing the Listing:

The +LIST+ alter code is used to specify which of the tagged lines are to appear in a particular listing, the title of the listing, and to instruct the program to print the listing. The user specifies the appropriate identification number or character and the title of the listing in the +LIST+ code. The program prints the listing at the end of the document.

Format of Output

The format of the TEXT360 output can be described in three major divisions: (1) supplemental listings that can be generated by the program (e.g., table of contents, list of figures, etc.); (2) body text, including appendices and index; and (3) miscellaneous other information, some of which is system-generated information and error messages listed at the back of the document and at the end of the program.

Format of Supplemental Listings:

TEXT360 can create a table of contents and other supplemental listings of items selected by the user. These listings are printed following the system-generated information at the end of the document. The program formats the listings as follows:

- The most recent column format specified in the input text is used as the column format for all supplemental listings.

- The most recent page depth specified in the input text is used as the page depth for the listings, with the exception that if the depth is greater than 68, the program sets the listing depth to 68.

- The listing title is positioned by the program as a level 1 heading. The capitalization of the title is as it appears in the +LIST+ code.

- The capitalization of an entry in the supplemental listings is taken from the capitalization of the entry as it appears in the input stream. The program does not manipulate capitalization as it does for the headings in text.

- The indention of an entry is determined by the first identification number or character included in the +LIST+ code. If it is alphabetic, none of the entries is indented. If it is numeric, the entries are indented, as follows:

  The numbers 1 and 2 cause the entry to be printed with no indention, but the number 1 entries are preceded by a blank line. All other entries are single-spaced.

  The number 3 causes the entry to be indented two spaces.

  The numbers 4 through 9 cause the entry to be indented four spaces.

  If an alphabetic character is specified within a +LIST+ code that is started by a number [e.g., +LIST+(1,2,A)+ title], the entries designated by the alphabetic characters are indented four spaces.

- The text of an entry that overflows to a subsequent line is aligned with the preceding line.

- The page number of each entry is printed at the right margin. Periods are printed between the last word of the entry and the page number.

- The most recent title, subtitle, and odd-page foot specified in the input text are printed in the listing; however, no page number is printed.

Format of Text:

The pages that are formatted in the text portion of the document consist of the following fields: running head, running foot, left and right control fields, gutter, and the text body. The composition of the printed text page with control fields is shown in Figure 16.

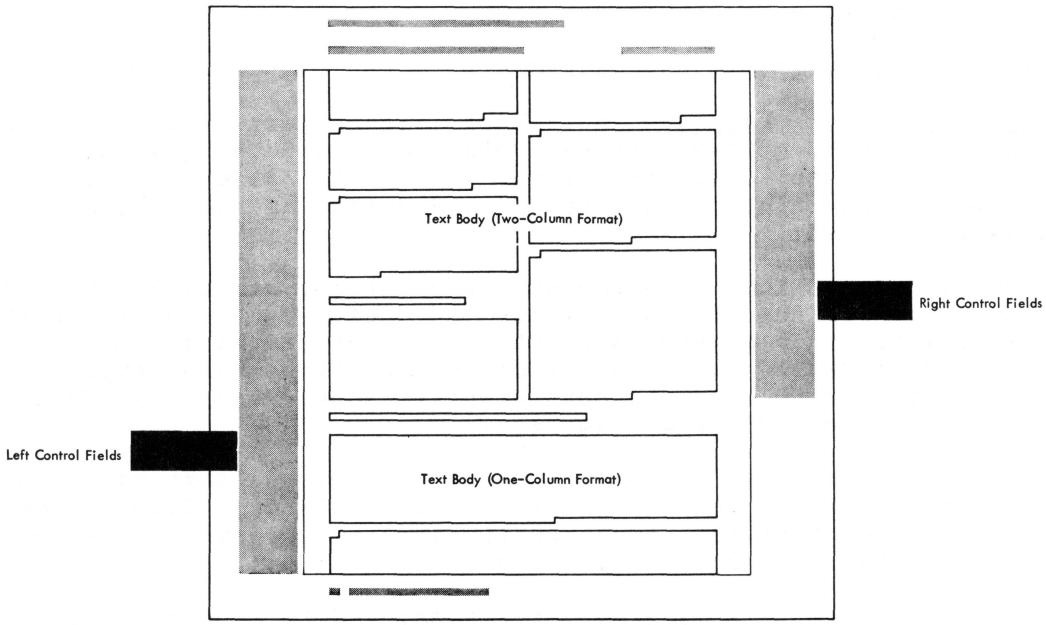


Figure 16. Format of Text Page

The running heads, running feet, and format of the text body are described in detail in a preceding section of this description; this section describes only the format of the control fields and the gutter.

CONTROL FIELDS: The control fields contain information necessary for making revisions. They are a total of 12 characters wide and are located at the extreme right and extreme left of the text body. Each field contains information concerning the adjacent text column. The printing of the control fields can be suppressed by the TEXT360 Print program (see "Appendix B: Peripheral Programs").

The format and content of the fields are controlled by the program. The left control fields are always used; the right control fields

are used only when printing in a two-column format. The control fields consist of the following:

- Line number or skip line indicator (three characters)
- Special blank indicator (#) (one character)
- Edit codes (six characters)
- Current revision marker (+) or program-generated blank line marker (•) (one character)
- Revision bar (|) (one character)

Line number or Skip Line Indicator: When a text line appears in the adjacent column, this field contains the number that the program assigns to the text line. It is this line number, combined with the page number, that provides a unique address for each line in the document.

The program assigns line numbers in sequence, starting with 001 for the first line of the single or left text column, with one exception: When a supplementary page is generated as a result of the overflowing of a logical sheet, the line number sequence does not revert to 001 for the supplementary page. The line numbering continues from the preceding page.

A line number is assigned to each line of one text column before continuing to the following text column. Although line numbers are assigned sequentially, occasionally they may appear to be out of sequence because of the following:

- Alter Code: A line number is assigned to each alter code that is retained on the master file at the point the alter code is encountered. Because the alter code is not printed in the text page where it is encountered (it is printed in the system-generated information), its omission causes an apparent gap in the sequence of line numbers.
- Two-column or Floating Keep: The program may move a two-column or floating keep from its input location, but the line numbers are assigned at the original location. This causes the line numbers to appear out of order. These line numbers are underscored for emphasis if the keep is moved to another page.
- Footnotes: The line number assigned to the line containing the footnote reference (the line preceding the footnote text in the input stream) and the line numbers of the footnote itself are not printed on the text page. Instead, the line number positions are filled with asterisks and the reference line and footnote lines, with their line numbers, are also printed in the system-generated information.

An "S" is printed in this control field in place of the line number when a skip line appears in the adjacent column. An S is also printed for all program-generated blank lines (e.g., blank lines following a heading) except for the two blank lines that accompany a two-column keep (this field is simply blank).

Special Blank Indicator: A pound sign (#) in this field indicates the presence of one or more special blanks (specified by /I) in the adjacent text line.

Edit Codes: The edit codes that occur within the adjacent text line are printed in this field. Under certain conditions, the edit-code field does not contain all of the edit codes specified in the text line. When such a condition exists, a partial representation of the edit codes is printed, and the entire line (text and edit codes) is printed in the system-generated information. The partial representation is printed whenever one of the following occurs:

- When the edit-code field cannot contain all of the edit codes that control the text line, only the first five positions are used for edit codes. The sixth position contains an asterisk, indicating the overflow of the edit-code field.

- When an edit code that specifies column positions (e.g., -V-) controls the text line, the column numbers are not printed in the edit-code field.

- When a tab edit code (-T-) controls the text line, only the T appears in the edit-code field. If the text line is controlled by more than one tab code, two T's appear in this field.

If an edit code appears within the line, as opposed to the beginning of a line (e.g., tabbed text at column position 20), a blank precedes the code in the edit-code field.

Current Revision Marker or Program-Generated Blank Line Marker: A plus sign (+) is inserted by the program in this field to identify a line that has been changed in the most recent revision. Every line of an initial run receives a plus sign. TEXT360 does not mark deletions.

A bullet (•) is inserted by the program in this field to identify the blank lines that the program inserts to perform column justification.

Revision Bar: The presence of a program-generated revision bar (|) in this field indicates that the adjacent text line has been updated since the last +NEWLEVEL+ alter code was entered.

The column position in which the bar is printed is controlled by the column format. It is printed immediately before the leftmost position of the column for a left or single text column, and

immediately after the rightmost position of the column for a right text column.

A revision bar is printed beside every line of an initial run.

The format of the control fields is illustrated in Figure 17.

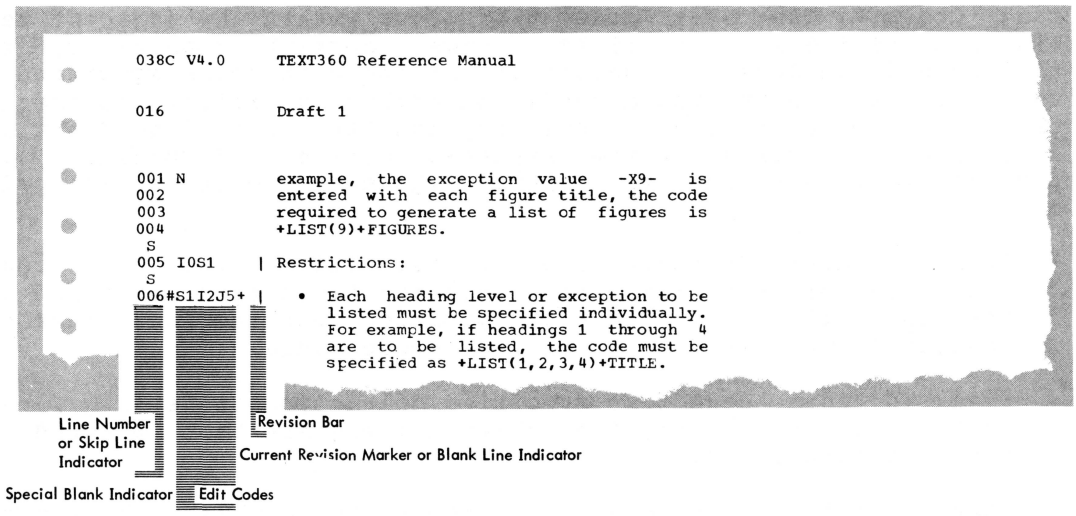


Figure 17. Control Fields

GUTTER: The gutter is the blank area separating the text columns when printing in a two-column format. It is a minimum of three characters in width and is centered on the text page. The width of the gutter can vary, depending upon the margin and line length settings.

Format of System-Generated Information and Error Messages:

The system-generated information and error messages are produced by the program to assist the user. The format and contents of the listings are controlled by the program.

ALTER CODES, EXPANDED EDIT CODES, AND FOOTNOTES: This listing can contain four distinct types of entries. When applicable, it contains every alter code retained in the master file and its location; every line that has a partial representation in the edit-code field; and every footnote line and its location.

Any edit code that overflows or is partially represented in the edit-code field causes an entry in this listing. The program lists the complete edit code and the text line in which it appears.

In addition to the footnote lines, the program also lists the text line that contains the footnote reference (the line preceding footnote text in the input stream). If a footnote is embedded within a text line, the portion that precedes and the portion that follows the footnote are assigned a separate line number.

ERROR MESSAGES: For quick recognition of error messages, the error messages are printed in a separate listing. The error message informs the user of the type of error, the corrective action (if any) taken by the program, and the location at which the error occurred. The location can be the page and line number in the new document, the page number of the document that is being updated, or an input line number.

If the error is such that the program can correct it (e.g., a column width specification that exceeds the maximum), the page and line number in the new document is given. If the error cannot be accepted by the program, the input line number or the old page number is given.

The TEXT360 error messages are listed and discussed in "Appendix H; TEXT360 Error Messages."

+SEARCH+ LINES: This listing is created whenever the user requests that the document be searched for a specific word or phrase [see "+SEARCH+ (Text Search)"]. For each occurrence of the text searched for, the listing contains the page number, the line number, and the complete text line containing the text in question.

+REVISE+ LINES: Whenever text replacement is performed by use of the +REVISE+ alter code [see +"REVISE+ (Text Replacement)"], the program generates this listing. The listing contains the page and line number, and the new text of each replacement.

SPELLING CHECK MESSAGES: TEXT360 can be instructed to compare the words of the input text with a dictionary (see "TEXT360 Spelling Dictionary Program"). If the input word is not found in the dictionary, the program prints the word and its associated page and line number in this listing.

DOCUMENT LINE UPDATING

Once a document has been produced by TEXT360, the user can delete and change lines, insert lines, insert portions of a line, delete and change controls, and copy text from one location to another with updating controls.

To perform updating that affects a particular location in a document, the user addresses the location by its page and line number. In the following explanations of codes, the page and line number is shown as

"ppplll" where "ppp" represents the page number and "lll" represents the line number.

Leading zeros can be omitted from the page number field. For example, "5007" refers to line 7 of page 5.

If more than one consecutive code affects a page, the page number may be omitted from the second and subsequent codes addressing that page. For example, "8100", "101", and "102" all refer to page 8 if entered in that order. The one exception to this rule is that of an updating code that is part of, or immediately follows, a "copy" function code [see "+ppplllC+ (Copying Text)"].

If a page number is omitted, leading zeros can be omitted from the line number; otherwise, they must be included. For example, "10002", "3", and "4" all refer to page 10 if entered in that order.

Restrictions:

- The updating controls must be entered with page and line numbers in ascending order. (For example, if the codes are entered in ascending order from pages 1 through 12 and another error is discovered on page 4, the code required to correct this error must be entered at its appropriate location within the page 4 corrections.)

- When two or more corrections affect the same line, the corrections must be entered in order from left to right.

- When addressing a line of a supplementary page of a logical sheet, the supplementary number is omitted (e.g., 24 is the page number, not 24.1).

- When updating a line of a floating keep that has been moved to another page, the original page number should be used.

- A change to a hyphenated word must be specified with the line address of the first portion of the word. (For example, if the word "information" is misspelled in the input and is hyphenated by the program as "informa-toin", the code used to transpose the letters "oi" must address the line in which "informa-" appears.)

+ppplll,ppplll+ (Deleting or Replacing Lines)

A +ppplll,ppplll+ alter code causes text lines to be deleted or replaced. The code causes the deletion of all lines beginning at the first page and line address and ending at the second page and line address.

The format of the +ppplll,ppplll+ code is:

```
+ppplll[,ppplll]+
```

A line is defined as beginning with an edit code, alter code, or first complete (nonhyphenated) word, ending with the last word or second portion of a hyphenated word. Any text following this code is inserted in place of the deleted text.

For example, to delete all text starting at page 7, line 5, to and including page 11, line 33, the code is specified as:

```
+7005,11033+
```

If new text is to replace the deleted text, the code is specified as:

```
+7005,11033+new text
```

To delete a single line, the code is specified as +ppplll+. For example, line 5 of page 7 is deleted by specifying:

```
+7005+
```

To insert new text in place of the deleted text, the new text is entered immediately following the code. The new text, whether it is to replace one line or several lines, need not be the same length as the deleted text.

When an updating error is encountered, any new data in the updating code is inserted at the location the error is encountered. For example, if line 5 of page 7 is deleted and replaced by new text, and the next updating code is out of sequence, any new text in that updating code is inserted after line 5 instead of the specified location. Also, the program ignores the first portion (e.g., deletion of a line) of the erroneous code. This is true of all the updating codes.

A +ppplll+ code can also be used to delete or replace an edit code that appears at the start of a line, without affecting the remainder of the line. Replacement is accomplished by specifying the +ppplll+ code, immediately followed by the new edit code. To delete an edit code, the +ppplll+ code must be followed by two consecutive minus delimiters (--).

For example, to replace the edit code -P- on page 7, line 5, with the edit code -S2I3-, the user specifies:

```
+7005+-S2I3-
```

To delete the edit code -P-, the user specifies:

```
+7005+--
```

When a +ppplll+ code is used in this fashion, it must be followed immediately by the next updating code or by the end input delimiter

(/Ø), if the next updating code is on the line following. The program, effectively, looks for text following the code; if no text is specified, it assumes only the edit code is to be changed. Any characters, including blanks, that might be left between the new edit code and the next updating code are interpreted as being the text that replaces the entire line.

For this reason, in order to delete an entire line and replace it with an edit code, a blank must follow the new edit code. For example, to replace line 7 of page 5 with the code to begin a new page, the user specifies:

```
+5007+-N-Ø
```

Restriction:

- Portions of an edit code cannot be deleted or replaced using this code unless the entire code is addressed. For example, -S1- in the edit code -S1I2J5- cannot be deleted or replaced in this manner without specifying the complete code (e.g., +ppp111+-S1I2J5-).

+ppp111R+ (Replacing Characters)

A +ppp111R+ alter code replaces a single character (or consecutive characters), word, phrase, or edit code in the line specified by the page and line number with the new text following the code.

The format of the +ppp111R+ alter code is:

```
┌──────────────────────────────────┐
│      +ppp111Rold+new             │
└──────────────────────────────────┘
```

where "ppp111" indicates the page and line number of the text that is to be replaced; "old" indicates the text to be replaced; and "new" indicates the text that is to replace the old text. There is no limit to the length of the replacing text; everything up to the next control or end input delimiter (/Ø) is considered to be the new text, if there is no new text on the line following.

It is important to remember that edit codes are included in the comparison when replacing characters, and that the alphabetic characters of the edit codes are considered to be capital letters.

Restriction:

> The text specified by "old" must not exceed 116 characters, or the program will ignore the complete replacement code. The special control characters (/, $, @, and *) are not included in the 116-character maximum.

To replace text, the program compares the "old" portion of the code with the contents of the specified line, starting at the beginning of the line. The comparison includes all controls in the line.

When a comparison is made, the new text is exchanged for the old, exactly as it is specified in the code. The "new" portion replaces the exact character configuration specified by "old". The exchange includes all specified blanks.

Underscoring, however, is not part of the comparison. Although the program locates "old" whether or not it is underscored, it removes the underscoring along with the "old" text. For example, if in the phrase "the handle need not have a prone status" the words "need not" must be replaced with "cannot", underscoring does not have to be specified for "need not", but it must be specified for "cannot".

If the "old" portion of the code compares with more than one section of the line, only the first occurrence is changed. For example, if, to correct the line "performance informanion innate to the compiler", the code is specified as +pppllllRAN+AT, the corrected line will read "performatce informanion innate to the compiler". In this example, the code must be specified as +ppplllRNI+TI.

The presence of blank characters is extremely important in a word replacement. If the blank at either end of a word is specified in "old", it must also be specified in "new". For example, to replace the word "or" with the word "and/or" in the sentence "A linkage editor or the loader....", the code must be specified in one of the following ways:

+ppplllR␣OR+␣AND/ZOR

+ppplllR␣OR␣+␣AND/ZOR␣

Note: A blank does not precede the first word of a line; therefore one cannot be included when replacing the first word of a line. However, a blank does follow the last nonhyphenated word of a line, should it ever be necessary to include one to identify the old text.

The +ppplllRold+new code can be used in two ways to delete or replace an edit code. One method provides for deleting or replacing an edit code in its entirety while the other provides for deleting or replacing a portion of an edit code.

To delete or replace the entire edit code, the +ppplllRold+new code must specify the entire old and new edit codes with delimiters (-). For example, to change the edit code -P- to S2I4-, the code is specified as +ppplllR-P-+-S2I4-.

To delete or replace a portion of an edit code, the minus delimiters are not specified, unless they are replaced; the alphabetic letters within the code, however, must be specified as capital letters.

For example, to delete the hanging indention of 5 from within the edit code -S1I2J5-, the user specifies:

```
+ppplllR*J5+
```

In this example, because nothing is to replace the deleted portion, the code must be immediately followed by the next updating code or end input delimiter, if the next updating code is on the line following. In the same example, the hanging indention of 5 can be changed to 3 by specifying:

```
+ppplllR5+3
```

It is important to remember that edit codes are included in the comparison when replacing characters and that the alphabetic characters of the edit codes are considered to be capital letters.

+ppplllI+ (Inserting Text)

A +ppplllI+ alter code is used to insert text after a line. The text following this code is inserted after the line address specified by "ppplll".

The format of the +ppplllI+ alter code is

```
+ppplllI+new text
```

where "ppplll" is the page and line number after which the new text is inserted. If the specified line ends with a hyphenated word, the text is inserted after the complete word, not at the point of hyphenation; "new text" indicates the text to be inserted. There is no limit to the size of the new text, which includes everything up to the next updating code.

+ppplllC+ (Copying Text)

Text can be copied from any location on the old master file to a specified location in the document being generated by using the +ppplllC+ code. This does not cause the text to be deleted from the old master file.

The format of the +ppplllC+ alter code is:

```
+ppplll¹C(ppplll²,ppplll³)[...(ppplll,ppplll)]+
```

where "ppplll[1]" is the page and line number of the position immediately after which the copied text is to be moved; "ppplll[2]" is the page and line number of the beginning of the text that is to be copied; and "ppplll[3]" is the page and line number of the end of the text that is to be copied.

The copy code is an instruction that temporarily repositions the old master file. When the copy code is encountered, the program repositions the old master file at the first page and line number specified within the parentheses. The specified section of text is copied and the old master file is repositioned at its original location. The text on the old master file is in no way affected.

By repositioning the old master file, text can be copied from any location. For example, a paragraph can be copied from page 17 to page 4, e.g.,

```
+4003C(17035,17048)+
```

or from page 4 to page 17, e.g.,

```
+17035C(4004,4016)+
```

Remember that a copy does not affect the old master file. For example, if a paragraph is copied from page 17 to page 4, it will also appear on page 17. If the paragraph is no longer required on page 17, it must be deleted from that location.

Because the old master file is repositioned, the portion of text specified within the parentheses can be updated in the normal updating manner, even though it appears as if the updating codes are out of sequence.

For example, to copy lines 35 through 48 of page 17 from their original location to following line 3 of page 4, at the same time inserting new text in place of line 39 of page 17, then deleting line 8 of page 4, the user specifies:

```
+4003C(17035,17048)++17039+new text+4008+
```

The copy code can be expanded to cause sequential copies from various locations on the old master file. The expanded code is:

```
+ppplllC(ppplll,ppplll)...(ppplll,ppplll)+
```

Each pair of page and line numbers designating a section of text to be copied must be enclosed within a separate set of parentheses. The sections of text being copied must be specified in the order in which they are to appear at the new location. There is no limit to the number of page and line numbers to be included in the sequential copies.

Restriction:

- If more than one pair of page and line numbers follow the copy code, only the last can be updated.

Text can also be copied to a location following a text insertion. The insertion is accomplished by the +ppplllI+ code. Both the copy code and the insertion code address the same location. For example, if an insertion is to follow line 1 of page 5 and copied text is to follow the insertion, the codes are specified as:

+5001I+new text+5001C(ppplll,ppplll)+

Restrictions:

- Complete page and line numbers must be specified for the section of text to be copied and for any updating code that is a part of, or immediately follows, the copy function; leading zeros and page numbers cannot be omitted.

- Copy codes cannot be nested; that is, text cannot be copied to a location that is itself being copied to another location.

+CPYOFF+ (Ending the Copy Function)

The +CPYOFF+ alter code suppresses the copy function. It is required if the updating code following a copy operation is within the limits of the copy function, so that the text can be updated in its original location. When this is the case, the program must have a way of knowing whether the updating code affects the original text or the copied text. The distinction is made by use of the +CPYOFF+ code.

The format of the +CPYOFF+ alter code is:

```
┌──────────────────────────────────────┐
│              +CPYOFF+                │
└──────────────────────────────────────┘
```

For example, in the following codes:

+5001C(5010,5030)++5017+

+5001C(5010,5030)++CPYOFF++5017+

The first code causes the program to copy (after line 1 of page 5), lines 10 through 16, delete line 17, and then copy (following line 16) lines 18 through 30. When the copy is completed, the remainder of the page, from line 2 on, is processed exactly as it appears on the old master.

The second code causes the program to copy lines 10 through 30 after line 1 of page 5. Then, as the next updating operation, the program deletes line 17 from its original location later on in the page.

April 1977

DOCUMENT SEARCHING AND REVISING

The user may wish to locate each occurrence of a word or phrase within a document, or to replace a word or phrase with a new word or phrase each time the original occurs. These functions can be achieved by use of the +SEARCH+ and +REVISE+ alter codes.

+SEARCH+ (Listing Specific Text)

The +SEARCH+ alter code causes the program to list, in the system-generated information at the end of the document, every occurrence of a specific combination of characters in the document. The user can also search for alter and edit codes. The program lists the page and line number and the complete text line in which the text appears. The text must be specified exactly as it appears in the document. It must include capitalization where it occurs.

The format of the +SEARCH+ alter code is:

```
┌──────────────────────────────────┐
│          +SEARCHbtext+           │
└──────────────────────────────────┘
```

where "b" is a required blank character which must precede the text; and "text" represents the text that is to be searched for. The word or phrase must be specified exactly as it appears in text, including capitalization; however, underscoring is ignored and need not be included.

Since the search is character by character, any valid character can be included in "text" except the plus sign, which delimits the text searched for.

The +SEARCH+ code is not retained on the master file. It is effective from the point where it is encountered to the end of the document or until ended by a +SEARCHOFF+ alter code.

Restrictions:

- The maximum number of characters that can be specified for "text" is 31. This figure includes blank characters, but not the special characters that control capitalization and underscoring or the required blank character.
- The maximum number of combined +SEARCH+ and +REVISE+ codes that can be operative at one time is 64.

+SEARCHOFF+ (Suppressing Text Search)

The +SEARCHOFF+ code ends the text search started by a +SEARCH+ code.

The format of the +SEARCHOFF+ alter code is:

```
+SEARCHOFF␣text+
```

where "␣" is a required blank character which must precede the text; and "text" represents the text specified in a +SEARCH+ code.

The +SEARCHOFF+ code is not retained on the master file.

Restriction:

- The text portion of this code must be specified exactly as its counterpart in the +SEARCH+ code.

+REVISE+ (Replacing Specific Text)

The +REVISE+ code replaces a combination of characters with another throughout the text. It also causes the program to list, in the system-generated information at the end of the document, the page and line number and the complete line containing the new text. The user can also replace codes, but care must be taken to specify capital letters when replacing a portion of a code (e.g., *I3, *J5, etc.).

The format of the +REVISE+ alter code is:

```
+REVISE␣old+new
```

where "␣" is a required blank character which must precede the old text; "old" represents the existing combination of characters; and "new" represents the replacement text.

Since the replacement is character by character, any valid character can be specified for "old" except for the plus sign, which delimits the old text.

The +REVISE+ code is not retained on the master file. It is effective from the point where it is encountered to the end of the document or until ended by a +REVISEOFF+ code.

Restrictions:

- The maximum number of characters that can be specified for either "old" or "new" is 31. This figure includes blank characters, but not the special characters that control capitalization and underscoring or the required blank character.
- The program does not check the underscoring of the old word when making the comparison; however, the underscoring status of the old word is always used for the underscoring of the new word.
- If text is altered that spans across two logical sheets, the new text is spaced entirely on the second sheet.
- The maximum number of combined +SEARCH+ and +REVISE+ codes that can be operative at one time is 64.

+REVISEOFF+ (Suppressing Text Replacement)

The +REVISEOFF+ code ends the text replacement started by a +REVISE+ code. Only the "old" portion of the +REVISE+ code need be specified.

The format of the +REVISEOFF+ alter code is:

```
┌──────────────────────────────────────┐
│          +REVISEOFFb/old+            │
└──────────────────────────────────────┘
```

where "b/" is a required blank character which must precede the old text; and "old" represents the existing text being replacement by a +REVISE+ code.

This code allows text replacement to be restricted to one portion of the document. The code is not retained on the master file.

Restriction:

- "old" must be specified exactly as its counterpart in the +REVISE+ code.

Specification of +SEARCH+ and +REVISE+ Codes

Because these functions invoke a comparison, the proper specification of "text" and "old" is extremely important. Any blank character that appears in "text" and "old" is considered in the comparison. Words are normally separated by one blank character; line-justification blanks are not considered.

If the old text is entered under the control of the -A- code or contains special blanks, additional blanks may be required to be specified in the comparison. (The first blank character between the word +SEARCH or +REVISE and the old text is a delimiter; it is not a part of the comparison.)

If "text" or "old" is not set apart by blank delimiters when searching for a single word, these codes make a comparison of any occurrence of the specified characters. For example, +SEARCH␣@MTS@@+ yields the location of MTS's, "MTS", (MTS), MTS,, MTS, and MTS. If the code is specified as +SEARCH␣␣@MTS@@␣+, only "␣MTS␣" is located, in addition to the occurrence of MTS at the beginning of a line and at the beginning of a sentence, but not when it is the first word of a paragraph.

Because the entire document is treated as one long character string, +SEARCH␣␣@MTS@@␣+ does not locate any MTS that is immediately preceded by a control (e.g., -P-@MTS@@).

Underscoring need not be specified in the old text, since the text is located independent of underscoring. Because both the +SEARCH+ and the +REVISE+ operations require a significant amount of processing, they should be used with discretion.

+INPUT+ (Merging Master Files)

Two or more master files can be merged to create a new master file. The merging is accomplished by the use of the +INPUT+ alter code. This code performs the switching function between the two files. Once the switch has been made, updating codes can extract data from the file. A detailed example of merging files is shown in "Appendix E: Coding Examples."

The format of the +INPUT+ alter code is:

```
┌──────────────────────────────────┐
│      +INPUT[␣master]+            │
└──────────────────────────────────┘
```

where "master" is a master file FDname to be attached to the ALTMAST parameter.

Each time the +INPUT+ code is encountered, the program switches from the master file currently in use to an alternate master file. The purpose of the +INPUT+ code is to switch from one master file to the other. Once the switch is accomplished, the alternate master file (attached to the ALTMAST parameter) starts processing only after an updating code is encountered (even if only the replacement of a single character).

After the required amount of data is extracted from the alternate master file, another +INPUT+ code following the updating codes causes the program to switch back to the original master file (the one at which initial processing started). Initial processing starts at the master file specified by the OLDMAST parameter (see "Appendix A: TEXT360 Program Requirements").

If "master" is specified on the +INPUT+ code, that file is first attached to the ALTMAST parameter and is then used as the alternate master file. If "master" is omitted and TEXT360 is currently processing from the old master file (the file attached to the OLDMAST parameter), processing is switched to the file currently attached to the ALTMAST parameter. If "master" is omitted and TEXT360 is processing from the alternate master file, processing is switched back to the file attached to OLDMAST.

Restrictions:

- The program must not be in logical sheet mode while merging master files.
- A +ppplllI+ code must precede the +INPUT+ code in the text flow (i.e., +ppplllI++INPUT+).
- This code causes the master file that is made inoperative to reposition itself back at the beginning of the file. When the file is activated again by the +INPUT+ code, the user must reposition it to the position where processing is to continue (i.e., +ppplll, ppplll+).
- If the master file FDname contains any of the special control characters -, +, *, /, $, @, and _ (when backspacing is in effect), they must be represented by /S, /A, /X, /Z, /D, /Q, and /J, respectively.

ALTER CODES

Alter codes are delimited by the plus sign. In general, alter codes remain in effect from the point at which they are encountered to the end of the document, or until they are changed.

This section lists the TEXT360 alter codes in alphabetical order and describes their formats and restrictions. When the correct syntax of an alter code is shown, the number of "n's" in the code is the same as the maximum number of decimal digits allowable.

The updating alter codes are not listed in this section. For complete descriptions of the updating alter codes, refer to the sections "Document Line Updating" and "Document Searching and Revising."

+ASIS+ (Direct Text Entry)

The +ASIS+ code invokes the direct text entry feature. All input text is entered directly into the document without any input translation being performed.

The format of the +ASIS+ alter code is:

```
┌──────────────────────────────────────┐
│               +ASIS+                 │
└──────────────────────────────────────┘
```

All input is treated as literal text. Each input line in the input source stream begins a new line of text in the document.

Restrictions:

- The special control characters (+, -, *, /, @, $, and _) are treated as literal characters; hence, the capitalization, underscoring, and backspacing functions are inoperative.
- Upper- and lowercase text is entered directly as specified without any case conversion being performed.
- The +ASISOFF+ code is the only control code recognized when +ASIS+ is in effect.
- Any text following the +ASIS+ control code on the same input line is ignored.

+ASISOFF+ (Suppressing Direct Text Entry)

The +ASISOFF+ code suppresses the direct text entry feature. This is the default.

The format of the +ASISOFF+ alter code is:

```
┌──────────────────────────────────────┐
│              +ASISOFF+               │
└──────────────────────────────────────┘
```

Restriction:

- The +ASISOFF+ control code must start in column one of the input line.
- Any text following the +ASISOFF+ control code on the same input line is processed.

+BACKSPACE+ (Backspacing)

The +BACKSPACE+ code invokes the backspacing (overprinting) feature. When +BACKSPACE+ is in effect, the special control character underscore (_) causes the following character to be overprinted on the preceding character, e.g., "b_/Z" is printed as "⌀".

The format of the +BACKSPACE+ alter code is:

```
┌──────────────────────────────────────┐
│              +BACKSPACE+             │
└──────────────────────────────────────┘
```

Restrictions:

- When backspacing is in effect, a literal underscore in the input text must be specified as "/J".
- A single character may be overprinted only once. Thus, A_B_C is invalid.
- Only one backspace character may be used in succession. Thus, AB__CD is invalid (this may be legally specified by A_CB_D).
- An input text line cannot have more than 47 backspace characters.
- Backspacing is not permitted in the input text used for the +TITLE+, +SUBTITLE+, +DATE+, +OFOOT+, or +EFOOT+ codes.

+BACKSPACEOFF+ (Suppressing Backspacing)

The +BACKSPACEOFF+ code suppresses the backspace feature.

The format of the +BACKSPACEOFF+ alter code is:

+BLANK+ (Padding of the Input Line)

The +BLANK+ code causes TEXT360 to pad all input lines with a single blank.

The format of the +BLANK+ alter code is:

+BLANKOFF+ (Suppressing the Padding of the Input Line)

The +BLANKOFF+ code suppresses the padding of the input line with a single blank. This is the default.

The format of the +BLANKOFF+ alter code is:

+COLJUST+ (Column Justification)

The +COLJUST+ code invokes the column justification feature. Because TEXT360 includes column justification unless otherwise instructed, the +COLJUST+ code is required only to reinstate this feature after it has been suppressed by the +COLJUSTOFF+ code.

The format of the +COLJUST+ alter code is:

+COLJUSTOFF+ (Suppressing Column Justification)

The +COLJUSTOFF+ code suppresses the column justification feature.

The format of the +COLJUSTOFF+ alter code is:

```
┌──────────────────────────────────────┐
│            +COLJUSTOFF+              │
└──────────────────────────────────────┘
```

+CPYOFF+ (Ending the Copy Function)

See this topic under the heading "+CPYOFF+ (Ending the Copy Function)" in the section "Document Line Updating."

+CONVERT+ (Input Conversion)

The +CONVERT+ code causes TEXT360 to convert all input text to lowercase before processing it. This is the default.

The format of the +CONVERT+ alter code is:

```
┌──────────────────────────────────────┐
│             +CONVERT+                │
└──────────────────────────────────────┘
```

+CONVERTOFF+ (Suppressing Input Conversion)

The +CONVERTOFF+ code suppresses the conversion of the input text to lowercase. When +CONVERTOFF+ is in effect, text may be entered directly in upper- and lowercase without using the special characters asterisk (*) and at sign (@) for capitalization.

The format of the +CONVERTOFF+ alter code is:

```
┌──────────────────────────────────────┐
│            +CONVERTOFF+              │
└──────────────────────────────────────┘
```

+DATE+ (Printing the Date)

The +DATE+ code specifies the date to be printed at the top of each text page; the date is supplied by the user. The date is printed on the same line as the subtitle, three lines below the title and next to the fold of the book (i.e., rightmost on even-numbered pages and leftmost on odd-numbered pages).

The format of the +DATE+ alter code is:

```
┌──────────────────────────────────────┐
│            +DATE+x...x               │
└──────────────────────────────────────┘
```

| where "x...x" represents the text of the date to be printed, which
| continues to the next control. The length of the text can be up to the
| page width and is considered to be "as-is" text.

+DEFINE+ (Multioperation Format)

The +DEFINE+ code allows a definition of a multioperation format function.

The format of the +DEFINE+ alter code is:

```
┌──────────────────────────────────────┐
│        +DEFINE?nn=x...x+             │
└──────────────────────────────────────┘
```

where "?nn" is the defined code. "nn" must be a number from 1 to 15; leading zeros may be omitted. Once specified, -?nn- can be entered into text as an edit code. "x...x" represents the individual edit codes that constitute the multioperation format function. The only controls that may be specified by the "x...x" are:

```
-A-     -I-     -S-
-C-     -J-     -X-
-D-     -K-     -Y-
-H-     -N-      $
         @
```

When specified for "x...x", these controls are enclosed by one set of delimiters.

With the exception of "H", these controls perform the same function whether included with the +DEFINE+ code or specified individually. The presence of "H" determines when capitalization and underscoring are to end if they have been specified.

When "H" is specified, capitalization and underscoring continue until the next control is encountered by the program. When "H" is omitted, capitalization and underscoring continue until a colon or period is encountered by the program. This means the user can simulate headings 2 through 4 by using "H", and headings 5 and 6 by omitting it [see "-H-(Headings)" in the section "Edit Codes"].

The +DEFINE+ code can contain two -S- codes. When two -S- codes are used, the first one entered becomes the skip preceding the text, and the second -S- code becomes the skip following the text. The inclusion of an -I-, -J-, or -S- edit code in a +DEFINE+ code does not override its value established previously by its use.

Example:

A heading can be defined by specifying +DEFINE?1=NS0@$HS3X2+. Subsequently, whenever -?1- is encountered, the program:

- Starts a new page.
- Skips no lines before the heading (which is necessary to include because of the skips after the heading).
- Prints the heading (the text from -?1- to the next control) in capital letters and underscored.
- Skips 3 lines after the heading.
- "Tags" the line with a value of 2. This allows the heading to be an entry in a program-generated listing such as the table of contents.

Restrictions:

- The -I-, -J-, -S-, and -X- codes must be specified with a parameter when used in the +DEFINE+ code. The -X?n- form of the -X- edit code may not be used.
- The +DEFINE+ code affects only the text following its definition. For example, if a multioperation format is defined at page 40, its corresponding -?nn- edit code cannot be entered into text before page 40 during subsequent runs.

+DEPTH+ (Page Depth)

The +DEPTH+ code specifies the page depth of the document being produced.

The format of the +DEPTH+ alter code is:

```
┌──────────────────────────────────────┐
│             +DEPTHnn+                │
└──────────────────────────────────────┘
```

where "nn" indicates the number of lines to be printed in a column, excluding the 10 lines used for running heads and feet. This number may be from 25 to 75. If no depth is specified, a depth of 50 lines is assumed.

Restriction:

- A new depth specification should precede a new width specification (+SWIDTH+, +DWIDTH+), but follow a new page specification (-N-), when it is specified together with these codes.

+DOUBLE+ (Double-Spaced Text)

The +DOUBLE+ code causes text to be double-spaced. Extra blank lines added by the program are counted when the depth of the page is calculated. Therefore, double-spacing reduces the amount of text of a page to about one-half the amount of a single-spaced page.

This is in contrast to the double-spacing facility of the TEXT360 Print program, which keeps the regular page and line numbers but prints the text double-spaced.

The format of the +DOUBLE+ alter code is:

```
┌──────────────────────────────────────┐
│              +DOUBLE+                │
└──────────────────────────────────────┘
```

The +DOUBLE+ code affects only the text of a document. Supplemental listings and system-generated information printed at the end of the document are printed single-spaced even if a +DOUBLE+ code is inserted in the text.

+DWIDTH (Two-Column Text)

The +DWIDTH+ code causes text to be printed in a two-column format and establishes the column width.

The format of the +DWIDTH+ alter code is:

```
┌──────────────────────────────────────┐
│             +DWIDTHnn+               │
└──────────────────────────────────────┘
```

where "nn" represents the column width in characters. The column width for a two-column format may vary from 20 to 52 characters. If no column format is specified, a one-column format with a column width of 65 characters is assumed.

A +DWIDTH+ code encountered part-way down the page causes the text preceding it to be formatted in one column (if the +DWIDTH+ code changes an +SWIDTH+ code) or two columns (if the +DWIDTH+ code is redundant) at the top of the page, and the text following the new code to be formatted in two columns at the bottom of the page. Two blank lines are inserted to separate the two formats.

Restrictions:

- The column width must always equal or exceed the line length [see "+LENGTH+ (Right Margin)"]. If the column width is set to less than the current line length, the program resets the line length equal to the column width.

- There must be space in the page being formatted for at least five lines of text per column when a column width or column format change is encountered by the program. If the space is not available, the program starts a new page. Two blank lines are inserted to separate the different formats. These two lines are taken from the five required and could result in only three lines of text being printed at the bottom of the page.

- Care must be taken when a change is made to a wider-width format in conjunction with a level 1 heading. -H1-+DWIDTH+ causes seven skips after the heading; whereas +DWIDTH+-H1- causes the title, subtitle, foot, and page number of the preceding page to be formatted as if the page were in the wider-column format. To circumvent this problem the +DEFINE+ code can be used for the heading, in which three skips are specified after the heading.

+EFOOT+ (Running Feet on Even-Numbered Pages)

The +EFOOT+ code specifies the running foot to be printed at the bottom of all even-numbered (left-hand) pages. If the program is instructed to lay out all text as right-hand pages [see "+FRONT+ (Odd-Numbered Pages)"], this foot is not printed, although it is retained on the master file and does not have to be respecified for a subsequent run.

The format of the +EFOOT+ alter code is:

```
┌──────────────────────────────────┐
│          +EFOOT+x...x            │
└──────────────────────────────────┘
```

where "x...x" represents the text of the running foot, which continues to the next control. The maximum length of the running foot is equal to the page width.

+FRONT+ (Right-Hand Pages)

The +FRONT+ code causes the title, subtitle, date, page number, running foot, and all level 1 headings to be formatted as on right-hand pages.

The format of the +FRONT+ alter code is:

```
┌──────────────────────────────────────┐
│              +FRONT+                 │
└──────────────────────────────────────┘
```

The +FRONT+ code is retained on the master file.

Restriction:

- If +EFOOT+ is specified while the +FRONT+ code is operative, the even-numbered foot is not printed; the program prints the text specified by +OFOOT+ on all pages. However, the text specified by the +EFOOT+ code is retained on the master file and need not be respecified for subsequent runs that omit +FRONT+.

+HEAD+ (Heading Sets)

The +HEAD+ code is used to select one of the two heading sets provided by TEXT360 (see Figure 6).

The format of the +HEAD+ alter code is:

```
┌──────────────────────────────────────┐
│              +HEADx+                 │
└──────────────────────────────────────┘
```

where "x" specifies the heading set to be used by the program. "x" can be either "A" or "B".

When a heading set is not specified by the user, heading set A is assumed.

April 1977

+HYPHALT+ (Adding Words to the Hyphenation Dictionary)

The +HYPHALT+ code is used to build a local dictionary of correctly hyphenated words which have been incorrectly hyphenated by the program.

The format of the +HYPHALT+ alter code is:

```
+HYPHALT+word[,...,word]¢
```

where "word" is a correctly hyphenated word.

Each word except the last must be followed by a comma. The last word must be followed by a blank character. If words overflow to a second line, the normal free-form input procedure applies. Only hyphens need be indicated; capitalization, underscoring, and other forms of punctuation need not be.

The hyphens within the words are indicated by "/S". For example, to specify the correct hyphenation of the word "information", the required code is:

```
+HYPHALT+IN/SFOR/SMA/STION¢
```

This code can be specified anywhere in the document. It becomes effective immediately and remains effective until the dictionary is cleared by the +HYPHCLR+ alter code.

Words entered into the local hyphenation dictionary are saved on the new master file.

Restrictions:

- The words to be entered in the dictionary must not be more than 24 characters, not including hyphens.
- The comma, minus sign, and blank are the only characters that cannot be specified within a word.
- The maximum number of words that the dictionary can hold is 40, including the words added by +HYPHCLR+. Note, however, that the HYPHDICT global dictionary may handle up to approximately 125,000 words.
- A word must be entered in the exact spelling of the word that is incorrectly hyphenated (e.g., "statements" will not necessarily be hyphenated correctly if only "statement" is entered in the dictionary).

+HYPHCLR+ (Creating a New Hyphenation Dictionary)

The +HYPHCLR+ code clears from the local hyphenation dictionary all previously specified words and enters into a new local dictionary the properly hyphenated words that follow the code. This code may appear to perform the same function as +HYPHALT+; +HYPHALT+, however, can add words to an existing dictionary, whereas +HYPHCLR+ starts a new dictionary of words each time it is specified.

The format of the +HYPHCLR+ alter code is:

```
┌──────────────────────────────────────┐
│    +HYPHCLR+word[,...,word]¢         │
└──────────────────────────────────────┘
```

where "word" is a correctly hyphenated word.

Each word except the last must be followed by a comma. The last word must be followed by a blank character. If words overflow to a second line, the normal free-form input procedure applies. Only hyphens need be indicated; capitalization, underscoring, and other forms of punctuation need not be.

The hyphens within the words are indicated by "/S". For example, to specify the correct hyphenation of the word "specialist", the required code is:

```
+HYPHCLR+SPE/SCIAL/SIST¢
```

This code can be specified anywhere in the document. It becomes effective immediately and remains effective until the dictionary is cleared.

Restrictions:

- The words to be entered in the dictionary must be more than 24 characters, not including hyphens.
- The comma, minus sign, and blank are the only characters that cannot be specified within a word.
- The maximum number of words that the dictionary can hold is 40, including the words added by +HYPHALT+.
- A word must be entered in the exact spelling of the word that is incorrectly hyphenated (e.g., "accounts" will not necessarily be hyphenated correctly if only "account" is entered in the dictionary.)
- The global HYPHDICT dictionary is not cleared.

+HYPHEN+ (Hyphenation)

The +HYPHEN+ code invokes the hyphenation feature. Because TEXT360 includes hyphenation unless otherwise instructed, the +HYPHEN+ code is required only to reinstate this feature after it has been suppressed by the +NOHYPH+ code.

The format of the +HYPHEN+ alter code is:

```
┌──────────────────────────────────────┐
│              +HYPHEN+                │
└──────────────────────────────────────┘
```

+IGNORE+ (Input Text Suppression)

The +IGNORE+ code causes the following input source lines to be ignored until the +IGNOREOFF+ code is encountered.

The format of the +IGNORE+ alter code is:

```
┌──────────────────────────────────────┐
│              +IGNORE+                │
└──────────────────────────────────────┘
```

The ignored text is not entered into the document and is not retained on the master file.

Restrictions:

- +IGNOREOFF+ is the only control code recognized when +IGNORE+ is in effect.
- The text that is ignored must be valid input to TEXT360; otherwise, the +IGNOREOFF+ alter code will not function properly.

+IGNOREOFF+ (Suppressing Input Text Suppression)

The +IGNOREOFF+ code suppresses the input text suppression feature.

The format of the +IGNOREOFF+ alter code is:

```
┌──────────────────────────────────────┐
│            +IGNOREOFF+               │
└──────────────────────────────────────┘
```

+INITIAL+ (Deleting Revision Bars)

The +INITIAL+ code eliminates all current and accumulated revision bars that are assigned by the program to indicate lines that have been changed.

The format of the +INITIAL+ alter code is:

```
┌──────────────────────────────────────┐
│              +INITIAL+               │
└──────────────────────────────────────┘
```

This code is normally used with an initial edition of a document.

+INPUT+ (Merging Master Files)

See this topic under the heading "+INPUT+ (Merging Master Files)" in the section "Document Searching and Revising."

+JUST+ (Line Justification)

The +JUST+ code invokes the line-justification feature. Because TEXT360 includes line justification unless otherwise instructed, the +JUST+ code is required only to reinstate this feature after it has been suppressed by the +NOJUST+ code.

The format of the +JUST+ alter code is:

```
┌──────────────────────────────────────┐
│               +JUST+                 │
└──────────────────────────────────────┘
```

+LENGTH+ (Right Margin)

The +LENGTH+ code establishes the right margin by specifying the column position in which the line must end.

The format of the +LENGTH+ alter code is:

```
┌──────────────────────────────────────┐
│            +LENGTHnnn+               │
└──────────────────────────────────────┘
```

where "nnn" indicates the column position of the right margin, which is measured from the leftmost position of the column, not from the current left margin setting. For example, +LENGTH50+ means that 50 characters, including blanks, are printed in the line. If a left margin is used, the 50 is reduced by the width of the margin. The number specified by the "nnn" must be from 20 to 107; leading zeros may be omitted.

The establishment of a right margin is referred to in this description as the "line length." The program assumes a line length of 65 characters if no +LENGTH+ code is specified. The default specification of 65 characters is set regardless of whether a +DWIDTH+ or an +SWIDTH+ code is specified by the user.

Restrictions:

- If a line length of less than 20 is specified, the program sets the length to 20.
- If a line length is specified which causes the value of the line length minus the left margin to be less than 20, the left margin is set by the program to the value which, when subtracted from the line length, leaves a remainder of 20.
- If a line length is specified greater than the column width, or, if the column width is set to a value less than the line length, the program sets the line length equal to the column width.
- Text controlled by the -A- edit code is limited by the full column width, not by the line length.
- If "nnn" is omitted, the program sets the length to 43.

+LIST+ (Supplemental Listings)

The +LIST+ code causes a listing to be created which is printed at the end of the document.

The format of the +LIST+ alter code is:

```
+LIST(x,...,x)+title
```

where "x,...,x" indicates the identification of the lines that are to be collected as entries in the listing; and "title" indicates the title of the listing. The title is positioned as a level 1 heading, but capitalization must be supplied by the user.

If the first character of "x,...,x" is numeric, the entries of the listing are formatted with various indentions (see "Format of Supple-

mental Listings" under "Format of Output"). If the first character is alphabetic, the entries are not indented. The format of the listing is the same as the format in effect at the end of the document, with the exception that the depth can be no greater than 68.

The following examples illustrate the application of this code.

- To generate a table of contents that lists headings 1, 3, and 4, the code is specified as +LIST(1,3,4)+*TABLE OF *CONTENTS.

- To generate a listing of figures, the figure titles must be assigned an exception value, which is assigned by tagging them with an -X- edit code. For example, if the exception value -X9- is entered with each figure title, the code required to generate a list of figures is +LIST(9)+*FIGURES.

Restrictions:

- Each heading level or exception value to be listed must be specified individually. For example, if headings 1 through 4 are to be listed, the code must be specified as +LIST(1,2,3,4)+title.

- No distinction is made between an exception value code (-X-) and a heading code of equal value (e.g., -X4- and -H4-). If 4 is specified within the +LIST+ code, both -X4- and -H4- entries appear in the listing.

- The maximum depth allowable per page for a supplemental listing is 68 lines; the maximum width allowable is 107 characters.

- A maximum of nine lists is allowed, each with a maximum of nine values.

- Capitalization desired in the supplemental listings must be supplied when the items are originally entered in the text stream.

+MARGIN+ (Left Margin)

The +MARGIN+ code establishes the left margin by specifying the column position in which the line must start.

The format of the +MARGIN+ alter code is:

```
┌──────────────────────────────────┐
│           +MARGINnn+             │
└──────────────────────────────────┘
```

where "nn" is the column position of the left margin, counted from the leftmost position of the column.

Restrictions:

- If a +MARGIN+ code is not specified, the program sets the margin to zero.
- If "nn" is greater than 87, the program sets the margin to 87.
- If the value specified by +LENGTH+ minus the value specified by +MARGIN+ is less than 20, the program sets the margin so that the difference equals 20.

+NEWLEVEL+ (Deleting Revision Bars)

The +NEWLEVEL+ code eliminates the accumulated revision bars that are assigned by the program to indicate lines that have been changed.

The format of the +NEWLEVEL+ alter code is:

```
┌──────────────────────────────────────┐
│              +NEWLEVEL+              │
└──────────────────────────────────────┘
```

Although the +NEWLEVEL+ code causes all previously accumulated revision bars to be cleared, revision bars are printed for the changes entered in the run in which +NEWLEVEL+ is specified.

+NOHYPH+ (Suppressing Hyphenation)

The +NOHYPH+ code suppresses the TEXT360 operation of hyphenation.

The format of the +NOHYPH+ alter code is:

```
┌──────────────────────────────────────┐
│              +NOHYPH+                │
└──────────────────────────────────────┘
```

+NOJUST+ (Suppressing Line Justification)

The +NOJUST+ code suppresses the operation of line justification.

The format of the +NOJUST+ alter code is:

```
┌──────────────────────────────────────┐
│              +NOJUST+                │
└──────────────────────────────────────┘
```

+OFOOT+ (Running Feet on Odd-Numbered Pages)

The +OFOOT+ code specifies the running foot to be printed at the bottom of all odd-numbered (right-hand) pages. If the program is instructed to lay out all text as right-hand pages [see "+FRONT+ (Right-Hand Pages)"], this foot is printed on all pages.

The format of the +OFOOT+ alter code is:

```
+OFOOT+x...x
```

where "x...x" represents the text of the running foot, which continues to the next control. The maximum length of the running foot is equal to the page width.

+PAGE+ (Suppressing Logical Sheet Mode)

The +PAGE+ code forces the program out of the logical sheet mode from the point where it is encountered to the end of the document. Out of the logical sheet mode, the program flows text from page to page without leaving blank pages or generating supplementary pages.

The format of the +PAGE+ alter code is:

```
+PAGE[nnn]+
```

where "nnn" specifies a starting page number.

If the number is greater than the number of the page being processed, a new page is started. The program numbers the new page with the number specified and continues sequential numbering from that point. For example, if +PAGE7+ is entered on page 2, the program ends the logical sheet mode and starts page 7 with the text that follows.

If the number is equal to the current page number or if "nnn" is not specified (+PAGE+), the program starts page numbering with the number of the page in process and does not start a new page.

If the +PAGE+ code specifies an even number, the program inserts a blank odd-numbered page before the even-numbered page requested.

The +PAGE+ code is not retained on the master file. It affects only the run in which it is specified. Unless the +PAGE+ code is specified, the program operates in the logical sheet mode.

For an initial run, a +PAGE+ code should precede all input text. If a +PAGE+ code is not specified, the program attempts to lay out the entire document as one logical sheet. After the first 999 lines, the program "repages" the document and flows text from page to page.

Restriction:

- The +PAGE+ code must specify a page number that is equal to or greater than the page being processed. The page number must not exceed three digits; leading zeros may be omitted.

+PRETDAY+ (Pre-T-Day Translation)

On February 22, 1988 MTS was changed to use the ISO EBCDIC coding standards. As a consequence, the codes used to represent some special characters were changed which necessitated that the affected files be translated.

If +PRETDAY+ is in effect, the following characters will have pre-T-Day hexadecimal representations for SCARDS input to TEXT360.

Character		029 Key Stroke	Hex. Rep.
]	Right bracket	/B	BD
[	Left bracket	/V	AD
}	Right brace	//B	9B
{	Left brace	//V	8B
~	Tilde	//T	73
^	Caret	//E	AA

The format of the alter code is

The +PRETDAY+ code should be specified for files that were created before T-Day and have not been translated.

+ppplll,ppplll+ (Deleting or Replacing Text)

See this topic under the heading "+ppplll,ppplll+ (Deleting or Replacing Text)" in the section "Document Line Updating."

+ppplllC+ (Copying Text)

See this topic under the heading "ppplllC+ (Copying Text)" in the section "Document Line Updating."

+ppplllI+ (Inserting Text)

See this topic under the heading "+ppplllI+ (Inserting Text)" in the section "Document Line Updating."

+ppplllR+ (Replacing Characters)

See this topic under the heading "+ppplllR+ (Replacing Characters)" in the section "Document Line Updating."

+REPAGE+ (Temporarily Suppressing Logical Sheet Mode)

The +REPAGE+ code is used to force the program temporarily out of logical sheet mode. When the program encounters this code, it flows text from page to page without leaving blank pages or generating supplementary pages.

The format of the +REPAGE+ alter code is:

```
┌──────────────────────────────────┐
│          +REPAGEnnn+             │
└──────────────────────────────────┘
```

where "nnn" is an even number that indicates the last page to be formatted before logical sheet mode is to be resumed. For example, if the code +REPAGE18+ is encountered on page 11, pages 11 through 18 are formatted without the control of logical sheet mode. If the text originally contained in this section does not fit in pages 11 through 18, the program starts generating supplementary pages for page 18. If there is not enough text to fill pages 11 through 18, the numbers of the empty pages are not used. The program does not provide blank pages.

The +REPAGE+ code is not retained on the master file. It affects only the run in which it is specified.

Restrictions:

- A +REPAGE+ code with no number supplied is an invalid code.

- Because a logical sheet consists of an odd-numbered page and its even-numbered backup page, the +REPAGE+ code should be entered on an odd-numbered page and specify an even-numbered page. If an odd-numbered page is requested, the program stops suppressing the logical sheet mode on the even-numbered page following the odd-numbered page requested.

- The +REPAGE+ code must specify a page number that is equal to or greater than the page being processed. The page number must not exceed three digits; leading zeros may be omitted.

+REVISE+ (Replacing Specific Text)

See this topic under the heading "+REVISE+ (Replacing Specific Text)" in the section "Document Searching and Revising."

+REVISEOFF+ (Suppressing Text Replacement)

See this topic under the heading "+REVISEOFF+ (Suppressing Text Replacement)" in the section "Document Searching and Revising."

Page Revised March 1988 April 1977

+SEARCH+ (Listing Specific Text)

See this topic under the heading "+SEARCH+ (Listing Specific Text)" in the section "Document Searching and Revising."

+SEARCHOFF+ (Suppressing Text Search)

See this topic under the heading "+SEARCHOFF+ (Suppressing Text Search)" in the section "Document Searching and Revising."

+SETTAB+ (Tab Settings)

The +SETTAB+ code specifies the tab settings.

The format of the +SETTAB+ alter code is:

```
+SETTAB[nn=nnn,...,nn=nnn[R]]+
```

where "nn" indicates the tab number; "nnn" indicates the column position in which text is to begin; and "nnnR" indicates the column position in which text is to end, right-adjusted. "nn" must be a number from 1 to 20; leading zeros can be omitted. "nnn" must be a number from 1 to 107; leading zeros can be omitted.

The program has default tab settings at 5-position intervals from column position 5 to 100 (i.e., 1=5,2=10,...,20=100). When the user specifies tab settings, the specified settings override the default settings or previously specified settings.

Only the specified settings are overridden. For example, if the user specifies:

+SETTAB1=3,2=9,3=38+

the default settings 4 through 20 are still operative.

If this code is then followed by +SETTAB1=4+, tabs 2 and 3 are still at the column positions specified in the previous +SETTAB+ code, and tabs 4 through 20 are at the default settings. To reset all tabs to the default settings, +SETTAB+ (without column positions) must be specified.

The column positions for the tab numbers need not be in ascending order (e.g., +SETTAB1=10,2=5+ is valid). +SETTAB1=100R+ causes the last character of text entered at that tab setting to be positioned in column 100. More than one right-adjusted tab setting is allowable.

Entering a large number of tab settings may cause the program to issue the error message "AS IS TEXT EXCEEDS 140 CHARACTERS." The program starts a new line with the excess characters, which then might cause a blank line to be generated, since the new line contains no text. To circumvent this problem, the user should enter fewer tab settings with more than one +SETTAB+ code. This can only occur if R is specified.

+SINGLE+ (Single-Spaced Text)

The +SINGLE+ code causes text to be single-spaced. Because TEXT360 includes single-spacing unless otherwise instructed, the +SINGLE+ code is required only to return from the double-spacing mode, specified by the +DOUBLE+ code.

The format of the +SINGLE+ alter code is:

```
┌──────────────────────────────────────┐
│              +SINGLE+                │
└──────────────────────────────────────┘
```

+SPELL+ (Spelling Comparison)

The +SPELL+ code causes words of the input text from 2 to 24 characters long to be compared with a dictionary for correctness of spelling. The dictionary is created by the TEXT360 Spelling Dictionary program (see "Appendix B: Peripheral Programs").

The format of the +SPELL+ alter code is:

```
┌──────────────────────────────────────┐
│              +SPELL+                 │
└──────────────────────────────────────┘
```

Words in the input flow that are not in the dictionary (including correctly spelled words not included) are listed with their page and line number in the spelling check message listing in the system-generated information. The code is not retained on the master file.

Restriction:

- The spelling check will be suppressed if a new master file is not specified.

+SUBTITLE+ (Printing a Subtitle)

The +SUBTITLE+ code specifies the text following it to be printed as the subtitle.

The format of the +SUBTITLE+ alter code is:

```
┌──────────────────────────────────┐
│        +SUBTITLE+x...x           │
└──────────────────────────────────┘
```

where "x...x" represents the text to be printed, which continues to the next control. This text is considered to be "as-is" text. The maximum length allowable is equal to the page width.

Restriction:

- When a date is printed, the maximum length of the subtitle is reduced by the length of the date. For example, if the page width is 80 and the date is 14 characters long, the subtitle can be 66 characters long. However, the program does not automatically provide spacing between the subtitle and the date. If a subtitle of 66 characters is specified, no blanks will appear between it and the date. Also, if a subtitle of more than 66 characters is specified, it will be overlaid by the date.

+SWIDTH+ (One-Column Text)

The +SWIDTH+ code causes the text to be printed in one column and establishes the column width.

The format of the +SWIDTH+ alter code is:

```
┌──────────────────────────────────┐
│          +SWIDTHnnn+             │
└──────────────────────────────────┘
```

where "nnn" indicates the column width in characters. The column width for a one-column format may vary from 20 to 107 characters. The default is +SWIDTH65+.

An +SWIDTH+ code encountered part-way down the page causes the text preceding it to be formatted in two columns (if the +SWIDTH+ code changes a +DWIDTH+ code) or one column (if the +SWIDTH+ code is redundant) at the top of the page, and the text following the new code to be formatted in one column at the bottom of the page. Two blank lines are inserted to separate the two formats.

Restrictions:

- The column width must always equal or exceed the line length. If the column width is set to less than the current line length, the program sets the line length equal to the column width.

- There must be space in the page being formatted for at least five lines of text per column when a column width or column format change is encountered by the program. If the space is not available, the program starts a new page. Two blank lines are inserted to separate the different formats. These two lines are taken from the five required and could result in only three lines of text being printed at the bottom of the page.

- Care must be taken when a change is made to a wider-width format (either +SWIDTH+ or +DWIDTH+) in conjunction with a level 1 heading. -H1-+SWIDTH+ causes seven skips after the heading; whereas +SWIDTH+-H1- causes the title, subtitle, foot, and page number of the preceding page to be formatted as if the page were in the wider-width format. To circumvent this problem the +DEFINE+ code can be used for the heading, in which three skips are specified after the heading.

+TDAY88+ (Post T-Day Translation)

On February 22, 1988 MTS was changed to use the ISO EBCDIC coding standards. As a consequence, the codes used to represent some special characters were changed which necessitated that the affected files be translated.

If +TDAY88+ is in effect (the default), the following characters will have post-T-Day hexadecimal representations for SCARDS input to TEXT360.

Character		029 Key Stroke	Hex. Rep.
]	Right bracket	/B	BB
[	Left bracket	/V	BA
}	Right brace	//B	D0
{	Left brace	//V	C0
~	Tilde	//T	A1
^	Caret	//E	B0

The format of the alter code is

```
+TDAY88+
```

The +PRETDAY+ code should be specified for files that were created before T-Day and have not been translated.

+TITLE+ (Printing a Title)

The +TITLE+ code specifies the title to be printed at the top of each text page.

The format of the +TITLE+ alter code is:

```
┌──────────────────────────────────┐
│          +TITLE+x...x            │
└──────────────────────────────────┘
```

where "x...x" represents the text to be printed, which continues to the next control. This text is considered to be "as-is" text. The maximum length allowable is equal to the page width.

Page Revised March 1988 April 1977

April 1977

EDIT CODES

Edit codes are delimited by the minus sign. In general, edit codes affect the format of a relatively small area of text.

This section lists the TEXT360 edit codes in alphabetical order and describes their formats and restrictions. When the correct syntax of an edit code is shown, the number of "n's" in the code is the same as the maximum number of decimal digits allowable.

Because many of the desired formats are obtained by a combination of individual functions, TEXT360 allows edit codes to be combined two ways. The user may enter edit codes fully delimited (e.g., -S1--I4--J8-) or in the more convenient, but equivalent, form of only one pair of delimiters (e.g., -S1I4J8-).

When the first method is used, the presence of a blank character between any two edit codes is critical; it can sometimes force a blank text line. For example, the combination -S1--I3- results in a skip of one line and an indention of three spaces for the next text line. If the code is entered as -S1-␣-I3-, the program interprets it as follows: -S1- causes a skip of one line; the blank causes a text line of one blank; -I3- starts a new line and indents the following text three spaces. Thus, the result appears to be that of -S2I3-.

In order to decrease the occurrence of ambiguous combinations of edit codes, the following rules apply:

- When two or more edit codes of the same kind (e.g., -S1S5-) are specified in combination, only the last code is used.

- Some edit codes are multioperation edit codes (e.g., -P- is the same as -S1I3-). When an edit code that performs an operation included in a multioperation edit code is specified with the multioperation edit code, the individually specified edit code overrides its predefined counterpart in the multioperation edit code. For example, if -S2P- is specified, the -S2- overrides the one-line skip included in the -P- edit code; the result is a skip of two lines and an indention of three spaces.

- When an -S- edit code is entered with an -H- or a +DEFINE+ code, the -S- code overrides the skip that precedes the heading or defined text (if one is included). The skip that follows the heading cannot be altered. (When consecutive headings are not separated by text, however, the program overrides the skip that precedes the second heading with the skip that follows the first heading.)

- If the user specifies an explicit skip before a heading code (e.g., -S2H3-), any keep in progress is not released. If the user specifies an explicit skip after a heading code (e.g., -H2S3-), or specifies the head any other way, any keep in progress is automatically

released. (Note: This applies to the order of specification, not to the skips preceding and following the heading itself.)

- When a combination code includes a code that starts a new line, the code that starts a new line should be specified first. For example, -B1,25A- and -AB1,25- do not yield the same result. -B1,25A- begins a table and then starts a new line with the text following the code. -AB1,25- starts a new line with the beginning of the table.

- When two or more edit codes that start a new line are specified in combination (e.g., -PJ4-), only one new line is started.

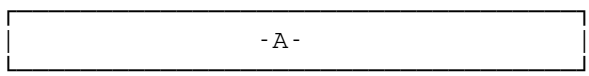
- The -Z- and -M- edit codes cannot be used in combination with other edit codes within one set of delimiters. When used with another edit code, they must be separated by at least one blank character (e.g., -Z-Ø-P-). The blank does not cause a blank text line.

- When an edit code that causes a horizontal line is used in combination with an edit code that causes a vertical line, the horizontal line code must be specified before the vertical line code to generate proper intersection.

-A- (Printing Text "As Entered")

Text following an -A- edit code is printed "as is" (the text is printed exactly as it appears in the input stream). The text controlled by an -A- code is formatted independently of other lines, and extra blanks within the line are not removed.

The format of the -A- edit code is:



The text started by this code ends at the next edit code that starts a new line. The "as-is" text prints to the full column width; it is not controlled by the line length.

Restrictions:

- The maximum number of characters that can be entered under the control of a single -A- code is 140. This number includes edit codes, blanks, and printable characters (i.e., *A, /S, and //V are all counted as one character each).

- The total number of printable characters, including blanks, must not exceed the column width. Otherwise, the program truncates the text at the column width.

April 1977

-B- (Beginning Tables)

A -B- edit code begins a table. It causes a horizontal line to be printed starting at the leftmost column position specified and ending at the rightmost column position specified. The horizontal line occupies one text line. This code also starts a vertical line in each of the column positions specified. The column positions need not be entered in ascending order. The vertical lines started by this code continue until an -E- code is encountered.

The format of the -B- edit code is:

```
┌──────────────────────────────────┐
│         -B[nnn,...,nnn]-         │
└──────────────────────────────────┘
```

where "nnn" specifies a column position.

A -B- code without column positions causes the program to use the positions specified in the most recent -B- code that specifies column positions.

A -B- code, combined with a -Bnnn,...,nnn- code to form -BBnnn,..., nnn-, causes the program to combine the column positions of the -BBnnn,...,nnn- code with those of the last -Bnnn,...,nnn- code. For example, if -B10,20,30- is followed by -BB5,15,40- the result is effectively -B5,10,15,20,30,40-. Subsequently, if only -B- is specified, it results in -B5,10,15,20,30,40-.

A subsequent -B- code can be specified while a previous -B- code is still operative. The subsequent code ends all vertical lines of the first before starting the new table. In this fashion, a table of varying format can be constructed by using the -B- code rather than horizontal and vertical-line coding.

It is not necessary to specify a -B- code at every text line to get the vertical lines. They need be specified only at the beginning of the table; they are generated by the program in each new line of text until ended by an -E- code.

A -B- edit code does not cause a new text line to be started.

Restrictions:

- If column positions are specified with the -B- code, there must be at least two; a -BB- code may specify one or more positions.

- A maximum of 22 vertical lines started by -B- or a -BB- code can be operative at a given time. This maximum does not include vertical lines started by the -V- code.

• All column positions must lie within the specified column width.

-C- (Centering Text Lines)

A -C- edit code causes the text following it to be centered. This code indicates the beginning of the text and starts a new line. The text ends at the next edit code that starts a new line. The centering is performed relative to the settings of the left margin (+MARGIN+) and the length of the line (+LENGTH+).

The format of the -C- edit code is:

```
┌──────────────────────────────────────┐
│                -C-                   │
└──────────────────────────────────────┘
```

Restrictions:

• All preceding and trailing blanks of the text are removed; all multiple-blank strings internal to the text are compressed into a single blank. The compression of internal multiple blanks may be suppressed by specifying the -C- code in combination with the -A- code (e.g., -S1AC-).

• Centering may not be used when drawing tables, vertical lines, or horizontal lines.

-D- (Beginning Two-column Keeps)

A -D- edit code indicates the beginning of a two-column keep.

The format of the -D- edit code is:

```
┌──────────────────────────────────────┐
│                -D-                   │
└──────────────────────────────────────┘
```

To define an area for supplemental copy (such as an illustration to be inserted later), the -D- code may be combined with the -S- code. For example, -DS20-*FIGURE 8./I/I*CAPTION-R- causes a skip of 20 lines, followed by the figure caption. If text is not entered between the -D- and -R- codes, the codes must be specified as -DS20-Ø-R-, not -DS20R- (or -DS20--R-). The embedded blank enables the -R- to end the -D- code action. The blank also causes one blank text line to be added to the 20 skip lines.

The program generates a two-line skip between a two-column keep and text, and between consecutive two-column keeps.

If any +LENGTH+ or +MARGIN+ changes are specified within a two-column keep, when the -D- code is released the line length and margin values revert to those in effect before the -D- code was encountered.

Restrictions:

- The program releases any portion of a two-column keep that exceeds the page depth.
- An -N-, -W-, -H1-, -K-, -Y-, or -F- code must not be specified within a two-column keep.
- If a -D- code is encountered while the program is processing in a one-column format, the -D- code is processed as a -K- code.
- Two-column keeps are printed at the top or bottom of the page, or of the segment of text that specifies the two-column keep if there is a change of column format in the middle of the page. However, if there is a level 1 heading on the page, the two-column keep is printed on the bottom of the page or segment of text.
- If a -D- code is encountered while the right column is being formatted and there is not enough room at the bottom of the page, the keep is printed at the top of the next page, and the remainder of the previous page is left empty.

-E- (Ending Tables)

An -E- edit code causes all vertical lines started by the preceding -B- edit code to be ended and a horizontal line to be printed between the leftmost and rightmost vertical lines of the table.

The format of the -E- edit code is:

```
┌──────────────────────────────────────┐
│                -E-                   │
└──────────────────────────────────────┘
```

Vertical lines started by a -V- edit code are not ended by this code.

An -E- edit code does not cause a new text line to be started.

-F- (Beginning Footnotes)

An -F- edit code indicates the beginning of footnote text. The footnote text should be entered in the input flow at the point of the footnote reference (i.e., immediately following the reference symbol or number, not at the end of the sentence). This is to assure that the footnote reference and the footnote itself will appear in the same column of text.

The format of the -F- edit code is:

```
┌──────────────────────────────────┐
│             ␣-F-                 │
└──────────────────────────────────┘
```

where "␣" is a required blank character which must precede the -F- code.

The -F- edit code is ended by a -Z- edit code.

Restrictions:

- A maximum of twenty footnote lines is allowed per column, including skip lines after the cutoff line. If the number of footnote lines exceeds 20, a new page is started with the current footnotes.
- Footnotes must not be specified within text controlled by a -D- code.
- An -H- code must not be specified within a footnote.

-G- (Horizontal Lines)

A -G- edit code causes one or more user-specified horizontal line segments to be printed.

The format of the -G- edit code is:

```
┌──────────────────────────────────┐
│     -Gnnn,nnn[,...,nnn,nnn]-     │
└──────────────────────────────────┘
```

where "nnn,nnn" specifies a pair of column positions. The first "nnn" specifies the column positions in which the line segment is to begin, and the second "nnn" indicates the column position in which it is to end.

The -G- edit code does not cause a new text line to be started.

Restrictions:

- All column positions must lie within the specified column width.
- The total number of column positions specified must be a multiple of two.
- A -G- code without specifying positions is invalid (it does not print the last horizontal line specified).
- The numbers in a pair of column-position specifications must be in ascending order, but the pairs need not be.

-H- (Headings)

An -H- edit code specifies a heading format of the selected heading set.

The format of the -H- edit code is:

```
┌──────────────────────────────────────┐
│                -Hn-                  │
└──────────────────────────────────────┘
```

where "n" specifies a number from 1 to 6, which indicates the specific heading format to be used. The headings, with their identifying numbers and format specifications, are shown in Figure 6.

A level 1 heading, which starts a new page, differs from the remaining formats in that it can span the full width of a page when printing in a two-column format. It is positioned at the top of the page according to the page number: right-adjusted if the page is odd-numbered and left-adjusted if the page is even-numbered. This right-adjustment may be overridden if the -H1- code is used in combination with the -C- edit code, e.g., -H1C-.

The other levels of headings are restricted to the width of the column, but may occupy more than one line of text.

There is an embedded -X- code of equal value with each -H- code specified for the purpose of listing a table of contents.

The format of a heading can be modified as follows:

- The number of skip lines that appear preceding the heading can be altered by specifying an -S- code with the -H- code. For example, -S1H2- causes the program to skip one line preceding the heading instead of two or three, depending on the heading set.

- When consecutive headings are not separated by text, the skip preceding the second and successive headings are overridden by the skips following the preceding headings.

Restrictions:

- -H- (without a value) is an invalid edit code.

- A level 1 heading is restricted to one line of text. It must not exceed the page width. (This restriction does not apply to heading levels 2 through 6, which can overflow to additional lines.)

- An -H- code must not be used within a footnote.

- The program capitalizes the headings according to the format specified in Figure 6, but they appear in supplemental listings (e.g., a table of contents) as they are specified in the text stream to appear in the listings.

- If the user specifies an explicit skip after a heading code (e.g., -H2S3-), any keep code in progress is automatically released.

-I- (Indentions)

An -I- edit code starts a new line and causes the first character of the line to be indented a specified number of spaces from the current left margin.

The format of the -I- edit code is:

```
┌──────────────────────────────────────┐
│              -I[nn]-                 │
└──────────────────────────────────────┘
```

where "nn" specifies the number of spaces the text line is to be indented.

If "nn" is specified as zero (-I0-), a new line is started with no indention. When "nn" is omitted (-I-), the program uses the "nn" value specified in the last nonzero -Inn- code. Leading zeros may be omitted in "nn".

Restriction:

- The indention must never cause the line length to be less than 20 characters. That is, the line length, minus the margin, minus the indention, must not be less than 20 characters.

-J- (Hanging Indentions)

A -J- edit code starts a new line and causes the second and subsequent text lines to be indented a specified number of spaces from the current left margin. This is known as a hanging indention. The program continues to indent each line thereafter until it encounters the next edit code that starts a new line.

The format of the -J- edit code is:

```
┌──────────────────────────────────────┐
│              -J[nn]-                 │
└──────────────────────────────────────┘
```

where "nn" specifies the number of spaces the second and subsequent text lines are to be indented.

If "nn" is omitted (-J-), the program uses the "nn" value specified in the last nonzero -Jnn- code. Leading zeros may be omitted in "nn".

Restrictions:

- The indention must never cause the line length to be less than 20 characters. That is, the line length, minus the margin, minus the indention, must not be less than 20 characters.
- A -J- code should not be specified within a line controlled by an -A- code.

-K- (Beginning Regular Keeps)

A -K- edit code indicates the beginning of a section of text that should not be divided between two columns or pages and which may not be moved from its original location in the input text. This is known as a regular keep.

The format of the -K- edit code is:

```
┌──────────────────────────────────────┐
│               -K-                    │
└──────────────────────────────────────┘
```

Restrictions:

- A keep cannot exceed the page depth. The program releases the -K- code at the point where the keep exceeds the depth. However, if a skip within the keep causes the keep to exceed the depth, the results are unpredictable.

- An -N-, -W-, or -H1- code should not be specified within a keep. If one of these codes is encountered while processing a -K- code, the program releases the -K- code and starts the action specified by the code.

- The -K- code does not start a new line. This feature allows a list and its introductory statement to be printed without being split by the program by embedding the -K- code in the line of text preceding the keep.

- If text is not entered between the -K- and -R- codes, a blank character must separate the codes (e.g., -KS20-Ø-R-). This blank also causes one blank text line to be added to the skip.

-L- (Horizontal Lines Within Tables)

An -L- edit code prints a horizontal line between the leftmost and rightmost column positions of the most recent -B- edit code. The intersections at the vertical lines are printed automatically by the program. The line occupies one text line.

The format of the -L- edit code is:

```
┌──────────────────────────────────────┐
│                -L-                   │
└──────────────────────────────────────┘
```

The -L- code does not cause a new text line to be started. It causes one continuous line to be printed, whereas the -G- code can cause a series of line segments to be printed.

-M- (Special Blanks with Pointers)

An -M- edit code causes the program to print a special blank (which the user can also specify with /I) in the text line in place of the -M- code and a pointer (< or >) in the margin or gutter next to the line containing the special blank.

This special blank is a "break" point and not a justification point. That is, no blanks are inserted for justification, but the line can be broken at this point, in which case the blank remains, either at the end of the line or at the start of the next.

The format of the -M- edit code is:

```
┌──────────────────────────────────────┐
│                -M-                   │
└──────────────────────────────────────┘
```

The pointer is printed in the gutter when formatting text in two columns and in the right margin when formatting text in one column.

Except for the insertion of a special blank and the pointer, the -M- code does not affect line formatting.

To delete or to insert text in place of the pointer, the updating code must specify the -M- code and not a blank (e.g., +ppplllR-M-+data). By deleting or replacing the -M- code, both the special blank and the pointer are automatically removed.

Restriction:

- This code cannot be combined within one set of delimiters with other edit codes. The -M- code must be specified individually and must be followed by at least one blank character when followed by another edit code (e.g., -M-␣-P-). The blank character does not cause a blank text line.

-N- (Beginning Pages)

An -N- edit code causes the text following it to begin a new page.

The format of the -N- edit code is:

```
┌─────────────────────────────────────┐
│                -N-                  │
└─────────────────────────────────────┘
```

To force the program to skip pages, this code must be entered in series with at least one blank character separating each -N- code. For example, -N-␣-N-␣-N- causes two blank pages; text begins on the third page.

When an -N- code is used in combination with an -S- code (e.g., -NS20-), the skip lines are not removed from the start of the new page, as other blank lines normally are.

Restriction:

- This code should not be specified within a regular, floating, or two-column keep, or within a footnote.

-P- (Beginning Paragraphs)

A -P- edit code causes the program to skip one line and indent the text following the code three positions from the current left margin. In other words, a -P- code is equivalent to -S1I3-.

The format of the -P- edit code is:

```
┌──────────────────────────────────────┐
│                -P-                   │
└──────────────────────────────────────┘
```

-Q- (Ending Vertical Lines)

A -Q- edit code ends vertical lines started by the previous -V- code.

The format of the -Q- edit code is:

```
┌──────────────────────────────────────┐
│          -Q[nnn,...,nnn]-            │
└──────────────────────────────────────┘
```

where "nnn" indicates a column position that has been specified in a -V- code.

When "nnn" is not specified, all vertical lines started by the previous -V- code are ended.

The vertical lines end in the text line in which the code is specified.

This code does not end vertical lines started by a -B- code; those vertical lines are ended by an -E- code.

-R- (Ending Keeps)

An -R- edit code releases -K-, -Y-, and -D- codes. The -R- code is specified immediately following the last character of the text to be kept.

The format of the -R- edit code is:

```
┌──────────────────────────────────────┐
│                -R-                   │
└──────────────────────────────────────┘
```

The text following the -R- code starts a new line when it ends a floating or two-column keep. When the -R- code ends a regular keep, the program does not start a new line.

An -R- code can be combined with edit codes that affect the text that precedes the release. For example, if the -R- code is to release a table, the -E- code which ends the table can be combined with the -R- code (e.g., -ER-).

Restrictions:

- When used in combination with other edit codes, the -R- code must be the last code in the combination (e.g., -AER-).

- An -R- code cannot be combined with edit codes that affect the text following the release. For example, if a new paragraph is to be started following a release, the -R- and -P- codes must be separated by at least one blank character (i.e., -R-␣-P-). The blank character does not cause a blank text line.

- Text must be specified between the code that begins the keep and the -R- code, even if it is only one blank character (e.g., -KS20-␣-R-). This blank also causes one blank text line to be added to the skip.

-S- (Skip Lines)

An -S- edit code causes the program to skip one or more lines before it prints the text following the code.

The format of the -S- edit code is:

```
┌──────────────────────────────────────┐
│              -S[nn]-                 │
└──────────────────────────────────────┘
```

where "nn" specifies the number of lines to be skipped, with the text being printed on the next line following the skip lines.

If -S0- is specified, a new line is started with no skip lines preceding it. If -S- is specified, the program uses the "nn" value specified in the last nonzero -S- code.

Under certain circumstances, skip lines are removed by the program. To ensure that the program does not remove skip lines that allocate space for supplemental copy (e.g., charts and figures), the skip must be specified in one of two ways:

- If skip lines are to appear at the start of a new column or page, or to allow space for a two-column figure, the -S- code should be specified in combination with the code required to obtain the new column, new page, or two-column keep (i.e., -W-, -N-, or -D- edit code). For example, to specify a skip of 20 lines at the start of a new column, the user specifies -S20W-. These skip lines are not removed by the program.

- If the skip is to allocate space for supplemental copy that is not to be formatted as described in the preceding paragraph, the skip lines are specified as -SnK-␣-Snn- or -SnY-␣-Snn-. The program interprets these codes as follows:

-SnK- (regular keep) or -SnY- (floating keep) represents the separator that is to appear between the supplemental copy and the text. This skip is ignored by the program if it appears at the start of a new column or page. If a separator is not desired, the user specifies -S0K- or -S0Y-.

The blank causes a blank text line to be printed. This blank line increases the total length of the skip by one line.

-Snn- specifies the space that is to be left for the supplemental copy. When this skip is encountered at the start of a new column or page, it is not ignored by the program, because it follows a blank text line.

Restriction:

- A skip request that exceeds the page depth produces a skip of one line and an error message (T360.106). The erroneous skip request is retained on the master file.

-T- (Tab Positions)

A -T- edit code causes the text following it to be printed at the tab setting specified.

The format of the -T- edit code is:

```
┌──────────────────────────────────────┐
│            -[.]T[nn]-                │
└──────────────────────────────────────┘
```

where "." causes periods to be printed in the tabbed-over space to the left of the text; and "nn" specifies a tab setting from 1 to 20, and which corresponds to an "nn" in the +SETTAB+ alter code.

-T-, without "nn", causes the number of the last tab setting to be increased by one and used. For example, -T4-*SUBTOTAL-T-*TOTAL causes the program to print "Subtotal" at the fourth tab position and "Total" at the column position specified for -T5-.

Restrictions:

- Tabbed text is treated as "as-is" text by the program, even when it is not specified with an -A- code.
- Tabbed text that exceeds the column width is truncated on the print file, but remains on the master file.
- -T0- is an invalid code; it is treated as -T1- without an error message being printed.

April 1977

-U- (Ending Tabular Text)

A -U- edit code allows tabular text and flowing text to be printed on the same line.

The format of the -U- edit code is:

```
┌──────────────────────────────────────┐
│                -U-                   │
└──────────────────────────────────────┘
```

The -U- code is entered in the input immediately before the text that is to flow. It indicates to the program that the tabular text has been processed and the program is to start formatting text in the regular manner. The -U- code does not start a new line.

The tabular and flowing text feature operates as follows. The margin is increased first (e.g., +MARGIN20+). This causes the normal printable area of the line to be decreased. Tab settings are then specified at column positions within the margin (e.g., +SETTAB1=3+). The text to be printed at this setting is specified and a -U- code is entered.

The program interprets the tab specifications, prints the tabbed text in the margin, and because the -U- code does not start a new line, the program prints the text following the -U- code in the area from the margin setting to the line length. The program continues to print in this portion of the column until another edit code alters the specifications.

When the -U- code is combined with the indention codes -I- and -J-, the indention codes do not start a new line.

The following examples illustrate the use of the -U- code to get tabular and flowing text.

(1) To print "NAME" in the left margin as shown in the example:

```
NAME Causes the program to read from
     storage ....
```

the example is entered:

```
+MARGIN11++SETTAB1=5+
-S1T1-@NAME-U-C@@AUSES THE PROGRAM TO READ
FROM STORAGE ....
```

(2) To print "NAME" in the right margin as shown in the example:

```
Causes the program to read     NAME
from storage ....
```

the example is entered:

```
+LENGTH31++MARGIN5++SETTAB1=36+
-T1-@NAME-U-C@@AUSES THE PROGRAM TO READ
FROM STORAGE ....
```

-V- (Beginning Vertical Lines)

A -V- edit code prints a vertical line in each column position specified.

The format of the -V- edit code is:

```
┌──────────────────────────────────┐
│          -Vnnn[,...,nnn]-        │
└──────────────────────────────────┘
```

where "nnn" specifies a column position in which a vertical line is to be printed. The column positions need not be entered in ascending order.

A -V- code (without column positions) does not cause the program to use the column positions of the most recent -Vnnn,...,nnn- code, if they have been ended by a -Q- code. However, a -V- code combined with a -Vnnn,...,nnn- code to form -VVnnn,...,nnn-, causes the program to combine the new column positions specified with those of the most recent -Vnnn,...,nnn- code that are still in effect. For example, if -V20,40- is followed by -VV5,15,30- (and there has been no intervening -Q- code), the result is effectively -V5,15,20,30,40-.

If a -V- code is specified while another -V- code of different specifications is operative, the second code ends the vertical lines in effect and starts new vertical lines at the positions specified.

It is not necessary to specify a -V- code at every text line to get the vertical lines. They need to be specified only at the beginning of the lines; they are generated by the program in each new line of text until ended by a -Q- code.

The -V- code does not cause a new text line to be started.

Restriction:

- A maximum of 22 vertical lines started by a -V- code can be in effect at a given time. This maximum does not include vertical lines started by a -B- code.

-W- (Beginning Columns)

A -W- edit code causes the text following it to be placed at the beginning of a new column.

The format of the -W- edit code is:

```
┌──────────────────────────────────────┐
│                -W-                   │
└──────────────────────────────────────┘
```

If the program encounters a -W- code while printing text in a one-column format, a new page is started. To cause the program to skip a column, the code must be entered in series with at least one blank character separating each -W- code (e.g., -W-␣-W-).

If the program encounters a -W- code in the left column while printing text in a two-column format, the new column is printed below any level 1 heading or two-column keep appearing at the top of the page.

When a -W- code is used in combination with an -S- code (e.g., -WS20-), the skip lines are not removed from the start of the new column, as other blank lines normally are.

Restriction:

- This code should not be specified within a regular, floating, or two-column keep, or within a footnote.

-X- (Specifying Text for Supplemental Listings)

An -X- edit code identifies text that can be collected by the program and printed with the corresponding page number in a supplemental listing.

The format of the -X- edit code is:

```
┌──────────────────────────────────────┐
│             -X[?]x-                  │
└──────────────────────────────────────┘
```

where "x" specifies either a number from 1 to 9 or an alphabetic character.

This code causes the text from the -X- edit code to the next edit code to be tagged with the value specified in the code. The corresponding page number of the text in the document is saved with the tag information for inclusion in the supplemental listings. The text of the edit code is included in the document.

If the edit code is specified in the form -X?n-, the text following the edit code is tagged for inclusion in the supplemental listings but is not inserted into the text of the document. In this manner, text may be included in the supplemental listings with its corresponding page number without being included directly in the document. In this case, the corresponding page number is the page number the text would have if it were included in the text of the document.

An -X- code starts a new line.

Restrictions:

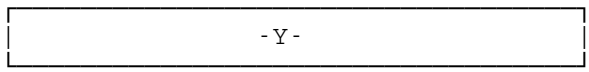
- Numeric codes 3 through 9 cause the text to be indented in the supplemental listing if the first code specified in the +LIST+ alter code is numeric; numeric codes 1 and 2 do not cause the text to be indented. Alphabetic codes cause the text to be indented four spaces if the first code in the +LIST+ alter code is numeric.

- Text that is specified with an -X1- code in conjunction with a code that starts a new page is positioned as a level 1 heading (i.e., right-adjusted on odd-numbered pages and left-adjusted on even-numbered pages). This right-adjustment may be overridden if the -X1- edit code is used in combination with the -C- edit code, e.g., -X1C-.

-Y- (Beginning Floating Keeps)

A -Y- edit code begins a section of text that cannot be divided between two columns or pages but can be moved from its original position within the input text. This is known as a floating keep.

The format of the -Y- edit code is:



The program may move the floating keep as much as one column from its original input location. This means that it can move across page boundaries. When a floating keep moves across page boundaries, line numbers assigned to the floating keep are underscored in the edit code list.

The movement of a floating keep is toward the end of the document. Although the floating keep can move with respect to the text following it, it is assigned page and line numbers at its original location within the input stream.

The principal purpose of a floating keep is to eliminate the gaps in text that can result when using a regular keep. A floating keep is moved in relation to the text following it. However, because a floating

keep cannot move past the next -Y-, -H1-, -W-, or -N- code, it is possible that the elimination of the extra space will not be achieved, if one of these codes should closely follow the keep.

The -Y- code starts a new line.

Restrictions:

- The program releases a -Y- code at the point where any text in the keep exceeds the page depth.
- An -N-, -W-, or -H1- code should not be specified within a keep. If one of these codes is encountered while processing a keep, the program releases the -Y- code and starts the action specified by the code.
- A floating keep cannot move past a -Y-, -H1-, -W-, or -N- code, nor can it move more than one column. A floating keep can move past a -K- (regular keep) code.
- If text is not entered between the -Y- and -R- codes, a blank character must separate the codes (e.g., -YS20-␣-R-). This blank also causes one blank text line to be added to the skip.

-Z- (Ending Footnotes)

A -Z- edit code ends the text designated as a footnote.

The format of the -Z- edit code is:

```
┌──────────────────────────────────┐
│             -Z-␣                 │
└──────────────────────────────────┘
```

where "␣" is a required blank character which must follow the -Z- code.

Restriction:

- This code cannot be combined within one set of delimiters with other edit codes. The -Z- code must be specified individually and must be followed by at least one blank character (e.g., -Z-␣-P-). The blank character does not cause a blank text line.

APPENDIX A: TEXT360 PROGRAM REQUIREMENTS

The TEXT360 program is invoked in MTS by the following command:

```
$RUN *TEXT360 SCARDS=input SERCOM=errors PAR=options
```

The logical I/O unit assignments are as follows:

SCARDS The TEXT360 source input.

SERCOM Error diagnostics.

The PAR field is used to specify the additional files and devices needed for the print file, master files, and other information produced by the TEXT360 program. These are specified as follows:

PRTOT=FDname The print file produced by TEXT360. The record format @U(132) is normally applied to this file. If omitted, PRTOT defaults to -PRINT.

MASTER=FDname The new master file produced by TEXT360. The new master file is required only if a future update is to be performed on the document or if a spelling check is to be performed.

OLDMAST=FDname The old master file to be updated. This is needed only if updating an old master.

ALTMAST=FDname Alternate old master to be updated. This is needed only if merging two master files.

EXCEPTN=FDname Exception file used to be used for generating the list of alter codes, expanded edit codes, and footnotes, the error messages, exception listings, and the +SEARCH+ and +REVISE+ lists. Since these are normally generated in virtual memory, a file should only be specified for this purpose when generating very large documents in order to reduce virtual memory charges.

HYPHDICT=FDname The global hyphenation dictionary file. This is used only if the +HYPHEN+ alter code is in effect (the default). The public file *HYPHDICT contains a hyphenation dictionary that may be assigned to HYPHDICT.

DICT=FDname The spelling dictionary file. This is used only if the +SPELL+ alter code is in effect. The public file *DICT contains a spelling dictionary that may be assigned to DICT.

When allocating file space for files, the following criteria should be used:

- The size of the print file (in pages) should be approximately equal to 1.1 times the number of text pages generated for the document.
- The size of the master file (in pages) should be approximately equal to 1.0 times the number of text pages generated for the document.
- The size of the exception file (in pages) should be approximately equal to 0.25 times the number of text pages generated for the document.

The cost of the resulting TEXT360 run will be less if these files are created to their proper sizes in advance.

TEXT360 performs its own blocking for the files attached to the PRTOT, MASTER, OLDMAST, and ALTMAST parameters.

The PRTOT print file is automatically written in the format @U(132) unless the file name begins with an "*" and is neither *SINK* nor *PRINT*, in which case PRTOT is assumed to refer to a magnetic tape. Print files may be written using unformatted (U), fixed-length (F, FB, or FBS), or variable-length (V, VB, or VBS) blocking formats.

For the MASTER, OLDMAST, and ALTMAST master files, a blocking factor of @U(256) is assumed unless specified on the FDname, e.g.,

```
MASTER=MASTFILE@U(4096)
```

If the FDname begins with an "*", it is assumed to refer to a magnetic tape. The tape record format may be used instead of @U(256). Master files may be read or written using unformatted (U) or variable-length (V, VB, VBS) blocking formats. Fixed-length blocking formats should not be used since extra blanks cannot be padded onto the end of master file lines.

When reading and writing print and master files from magnetic tapes, blocking is disabled for the magnetic tape DSR.

The print file is trimmed of trailing blanks when written. The master files are written untrimmed and must always remain untrimmed. Therefore, when master files are copied to other files or to magnetic tapes, they must be copied using the @¬TRIM FDname modifier.

The following table gives the approximate costs of running TEXT360. These costs are based on running the TEXT360 program on the section "TEXT360" in this volume (146 pages).

Type of TEXT360 Run	Cost
TEXT360 run which generates only a print file.	$ 9.63
TEXT360 run which generates a print file and a new master file.	10.95
TEXT360 run which generates a print file and a new master file and performs a spelling check.	12.24
Print program run which generates the printed output on a line printer (*PRINT*).	5.77

Thus, as can be seen from the above table, a normal production run of TEXT360 that produces only a print file which is to be written to the line printer will cost approximately (963+577)/146 = 10.5¢ per page. This estimate is based on March 1977 normal priority, University-Government rates.

The following example illustrates how to set up a TEXT360 run using line files for the source input, the old and new master files, and the print file.

```
$RUN *TEXT360 SCARDS=UPDATE PAR=PRTOT=PRINT MASTER=NEWMASTER
     OLDMAST=OLDMASTER HYPHDICT=*HYPHDICT
$RUN *PRINT SPRINT=*PRINT* 0=PRINT
A,1TOEND
$ENDFILE
```

This example reads update information from the file UPDATE to be applied to the old master file OLDMASTER and generates a new master file NEWMASTER. The print file is PRINT.

The following example illustrates how to set up a TEXT360 run using a line file for the source input and magnetic tapes for the old and new master files and the print file.

```
$MOUNT
C0000 9TP *O* VOL=OLDM 'OLD MASTER FILE'
C0001 9TP *M* VOL=NEWM RING=IN 'NEW MASTER FILE'
C0002 9TP *P* VOL=PRNT RING=IN 'PRINT FILE'
$ENDFILE

$CONTROL *O* POSN OLDMASTER
$CONTROL *M* POSN *EOT*
$CONTROL *P* POSN *EOT*
$CONTROL *M* DSN NEWM
$CONTROL *P* DSN NEWP

$RUN *TEXT360 SCARDS=UPDATE PAR=PRTOT=*P*@VB(13200,136)
     MASTER=*M*@VB(25600,256) OLDMAST=*O*@VB(25600,256)
     HYPHDICT=*HYPHDICT

$CONTROL *P* POSN NEWP
$CONTROL *P* BLK ON

$RUN *PRINT SPRINT=*PRINT* 0=*P*
A,1TOEND
$ENDFILE
```

For the print file, a blocking format of VB(13200,136) is used. Since the TEXT360 program performs its own blocking, it disables the blocking feature. However, the Print program does not perform its own blocking; therefore, blocking must be enabled.

APPENDIX B: PERIPHERAL PROGRAMS

Prescan

The Prescan program is designed to detect coding specification errors such as incorrect syntax, invalid parameters, and improper page and line sequencing. The use of this program can greatly reduce the cost of producing a document.

The Prescan program is invoked by the following command:

```
$RUN *PRESCAN SCARDS=input SPRINT=listing SERCOM=errors PAR=options
```

The logical I/O unit assignments are as follows:

SCARDS — The TEXT360 input to be scanned for errors.

SPRINT — The listing of the input plus error diagnostics, if the LIST option is specified.

SERCOM — The error diagnostics, if the ERR option is specified.

The following options may be specified in the PAR field.

MAJ=mm — The number of permissible major errors. If there are more than "nn" major errors, the program will return with a return code of 8. The default for MAJ is zero.

MIN=nn — The number of permissible minor errors. If there are more than "nn" minor errors, but not more than "mm" major errors, the program will return with a return code of 4. The default for MIN is 5.

ERR — If the ERR option is specified, the error diagnostics with the corresponding line numbers in the input file where the error occurred are printed on the logical I/O unit SERCOM. The default is NOERR for batch mode and ERR for conversational mode.

NOERR — The error diagnostics are not printed.

LIST — If the LIST option is specified, a listing of the TEXT360 input file, the corresponding MTS line numbers of the file, and the error diagnostics are printed on the logical I/O unit SPRINT. The default is LIST for batch mode and NOLIST for conversational mode.

NOLIST — A listing is not printed.

WARN — If the WARN option is specified, informational (warning) messages are included as part of the error diagnostics. The default is WARN.

NOWARN — Informational messages are not printed.

The input to the Prescan program can be either a TEXT360 initial input file or an update file. The output is a listing of the input, with error diagnostics, when applicable.

A perfect scan does not necessarily mean an error-free TEXT360 run, however, since certain errors cannot be detected. Likewise, an error detected by the Prescan program run may not actually cause an error in the TEXT360 run. For example, to delete an edit code the user may specify +pppll1+--/⌀; however, this also results in a Prescan error message which says that an invalid edit code has been entered.

The Prescan error messages are listed in "Appendix G: Prescan Error Messages" along with an indication of whether the message is a major error, minor error, or warning message.

TEXT360 Print Program

The TEXT360 Print program lists the print file generated by the TEXT360 program.

The TEXT360 Print program is invoked by the following command:

```
$RUN *PRINT SCARDS=options SPRINT=output SERCOM=errors
     0=input ...  PAR=options
```

The logical I/O unit assignments are as follows:

SCARDS The print instructions if the PAR field is omitted.

SPRINT The print output.

SERCOM Diagnostic error messages.

0-19 Input print files.

The print instructions (see below) may be specified in the PAR field or via the logical I/O unit SCARDS. If both SCARDS and a PAR field are specified, the PAR field will be used.

A series of print instructions may be given to the Print program which specify such things as which pages to print, double-spacing of the output, and printing in upper- or lowercase. The codes used to specify the print instructions and their functions are:

Codes	Function
#n	Use logical I/O unit "n", where "n" may be from 0 to 19 (0 is the default).
A	Use logical I/O unit 0.
B	Use logical I/O unit 1.
C or CHANGES	Print only changed pages (any individual pages specified are ignored).
D or DOUBLE	Print double-spaced format.
I or NOBARS	Suppress revision bars.
L or LEAVE	Do not rewind logical I/O unit.
M or NOMESSAGES	Suppress the listing of alter codes, expanded edit codes, and footnotes.
MCC	Print with machine carriage-control (default is logical carriage control)
N	Terminate the program.
P	Print in portrait mode (for Xerox 9700)
R or NOREVISE or NOSEARCH	Suppress the SEARCH and REVISE listings.
S or NOCONTROL	Suppress control fields.
U or UC	Print in uppercase.
X or NOSPELL	Suppress the SPELL listing.
mmmTOnnn	Print consecutive pages "mmm" to "nnn".
mmm-nnn	Same as above.
mmmTOEND	Print pages "mmm" to the end of the document.
mmm-END	Same as above.
nnn,...,nnn	Print individual pages. Each "nnn" indicates a page. Blanks instead of commas may be used to separate page specifications.

The codes of the print instruction may be separated by a comma or blanks.

For each print instruction, a maximum of 50 page-range specifications may be given. Thus, a user may specify at most 50 unique pages or any combinations of "mmmTOnnn" to be printed. For each print instruction, the program always scans the print file from the start to the end. Therefore, page numbers to be printed must be listed in ascending order and cannot be repeated. After the print instruction processing is completed, the Print program rewinds the print file, so that the same logical I/O unit may be used in successive print instructions. The L code may be used to prevent the rewinding of the logical I/O unit; this is useful only for print files that are contained on magnetic tapes, where the user may not desire to rewind the tape back to the load point.

The listing referred to by the M code is the listing of alter codes, expanded edit codes, and footnotes generated by the TEXT360 program which appears at the end of the print file. Specifying the M code does not suppress error messages.

The control fields referred to by the S code are the page-line and edit-code specifications which appear in the left margin for one-column text and in both the right and left margins for two-column text.

The C code is used in conjunction with updating master files. If the C code is specified, only pages which have been changed during the updating process are printed. A changed page is any page that contains a change specified by an update code or any page generated while the TEXT360 program is not in logical sheet mode. A change to a logical sheet causes its supplementary pages to be changed pages; however, the preceding pages are not changed pages. For example, if page 26 is a changed page, its supplementary pages (i.e., 26.1, 26.2, etc.) are also changed pages; page 25 is not, nor is page 27.

The changes made to changed pages are indicated by revision bars which appear next to the lines of text that have been revised. These revision bars may be suppressed by the I code.

For example, the print instruction

```
A,M,S,1TOEND
```

will read the print file from logical I/O unit 0 and print the entire document from beginning to end, suppressing the control fields and the listing of alter codes, expanded edit codes, and footnotes. The print instruction

```
#2,I,S,1,5,10-15,20-30
```

will read the print file from logical I/O unit 2 and print pages 1, 5, 10-15, and 20-30, suppressing the revision bars and the control fields.

| The output produced by the Print program is in logical carriage-
| control format.

TEXT360 Spelling Dictionary Program

The TEXT360 Spelling Dictionary program creates and maintains the dictionary required for the spelling check in the TEXT360 program.

The spelling check causes each word of the input text to be compared with the words in the dictionary. Words that are not in the dictionary (e.g., misspelled words) are listed in the spelling list of the user-reference material along with their associated page and line numbers. The dictionary size limitation and the fact that not all correctly spelled words are included causes some correctly spelled words to be listed.

The TEXT360 Spelling Dictionary program is invoked by the following command:

```
$RUN *DICTUPD SCARDS=input SPRINT=listing PAR=options
```

The logical I/O unit assignments are as follows:

SCARDS The input lines (see below).

SPRINT The listing output.

The PAR field is used to specify the files containing the current spelling dictionary and the new dictionary produced from the current dictionary and input lines. These are specified as follows:

DICT=FDname The name of the file containing the current dictionary. If this is omitted and is required, the user will be prompted for the file name. This file is not required if initializing a new dictionary.

NEWDICT=FDname The name of the file to be used for the new dictionary produced. This may be omitted if no new dictionary is being produced.

The input to the TEXT360 Spelling Dictionary program must be entered with the first column position indicating the function to be performed. Column 1 can specify one of the following codes:

Codes	Function
I	Initiate a new dictionary.
A	Add to the current dictionary.
D	Delete from the current dictionary.
R	Read the current dictionary.
P	Print the new dictionary.
¢	Use the most recent I, A, or D code specified (or use I if no code is specified).

When column 1 is left blank, the program uses the most recent I, A, or D code specified. This allows more than one line to be entered easily for more than one function.

The remainder of the input line (other than column 1) is used to enter the words that are to be acted upon by the program. The words must be separated by at least one blank character and, unlike the TEXT360 free-form input, words cannot be split between lines. The program accepts only words from 2 to 24 characters in length; words that do not meet this requirement are ignored by the program.

The spelling dictionary can hold a total of 75,241 words. The number of words allowed is dependent on the number of characters in each word The maximum number allowable for each word-length is as follows:

Words of length	Maximum number	Words of length	Maximum number
2	676	14	2340
3	10922	15	2184
4	8191	16	2047
5	6553	17	1927
6	5461	18	1820
7	4681	19	1724
8	4095	20	1638
9	3640	21	1560
10	3276	22	1489
11	2978	23	1424
12	2730	24	1365
13	2520		

The P code may be used to obtain a listing of the spelling dictionary. An update may be performed at the same time with the updated dictionary placed in the NEWDICT file. If none is required, two lines may entered, both of them blank except for column 1. The first line should contain an "R" in column 1, and the second line should contain a "P" in column 1.

The program prints the dictionary in alphabetical order according to the number of characters in the word (i.e., two-letter words in alphabetical order first, then three-letter words in alphabetical order, etc.).

For example, the following example

```
$RUN *DICTUPD PAR=DICT=OLDDICT NEWDICT=DICTIONARY
D VARIABLED
A VARIABLES
P
$ENDFILE
```

will read the current dictionary file OLDDICT, delete the word VARIABLED, add the word VARIABLES, and print the new dictionary file DICTIONARY

A source dictionary contained in the file *DICT is included with the TEXT360 program as a base for the user's dictionary. This dictionary may be added to, deleted from, or eliminated entirely, and a new dictionary created. This dictionary, or the user's own dictionary, may be referenced by inserting the +SPELL+ alter code in the TEXT360 input if a spelling check is desired. Note that the use of the spelling check feature causes a significant increase in the processing time for TEXT360; therefore, it should be used with discretion.

TEXT360 Global Hyphenation Dictionary

A global hyphenation dictionary may be used to specify the correct hyphenation of words incorrectly hyphenated by TEXT360. The global dictionary is maintained in the form of a file. This file is specified in the PAR field using the HYPHDICT parameter:

HYPHDICT=FDname The name of the file containing the current global hyphenation dictionary. This may be omitted if no global hyphenation dictionary is being used.

If the +HYPHEN+ alter code is in effect and a word is to be hyphenated by TEXT360, the program first checks the local hyphenation dictionary (the dictionary created by the +HYPHALT+ alter code); if the word is found, that hyphenation form is used. If it is not found, the program then checks the global hyphenation dictionary; if it is found, that hyphenation form is used. If it is not found, the word is hyphenated by the program.

The public file *HYPHDICT contains a list of words which are known to be hyphenated incorrectly by the program. This file can be used as the global hyphenation dictionary by specifying

HYPHDICT=*HYPHDICT

The user may also construct his own global hyphenation dictionary. Each word is entered in the dictionary in its hyphenated form using the minus sign (-) to indicate where the word may be hyphenated. More than one word may be given per line; words on each line must be separated by commas and/or blanks. Words may be entered into the dictionary in any order. Words may be from 2 to 24 characters in length.

The following three lines are taken from *HYPHDICT to illustrate the format of dictionary entries.

```
ac-com-plish, ac-com-plished, ac-com-plish-ing
ac-cord-ing, ac-cord-ing-ly
ac-knowl-edge, ac-knowl-edg-ment
```

The user may use his own global hyphenation dictionary and the *HYPHDICT dictionary by specifying

HYPHDICT=FDname+*HYPHDICT

where "FDname" is the file containing the user hyphenation dictionary.

April 1977

TEXT360 Master-to-Input Conversion Program

The TEXT360 Master-to-Input Conversion program converts a TEXT360 master file into a TEXT360 source input file. This program is invoked by the following command:

```
$RUN *T360MASTCONV SCARDS=input SPRINT=errors SPUNCH=output
     PAR=MASTER=FDname
```

The logical I/O unit assignments are as follows:

SCARDS The list of page-line number pairs to be included from the master file.

SPRINT Prompting and error messages.

SPUNCH TEXT360 source input file produced.

The PAR field is used to specify the file or device containing the TEXT360 master file to be converted. This is specified as follows:

MASTER=FDname "FDname" is the master file to be converted. A blocking factor may be specified with "FDname". If the file name does not begin with an asterisk "*" and if no blocking factor is specified, the blocking factor @U(256) is assumed.

The TEXT360 source input file produced will be contain both uppercase and lowercase letters; hence, the first line will be the alter code +CONVERTOFF+. The input file also assumes that the alter codes in effect are +BLANKOFF+, +BACKSPACEOFF+, and +ASISOFF+. If the +BACKSPACE+ alter code is to be used with the file, the user must change all occurrences of "_" to "/J" in the source input file produced; this may be done via the MTS editor using the edit command

```
ALTER@A /FILE '_'/J'
```

The +BLANK+ alter code should not be specified for the source input file; otherwise, blanks may be inserted into the middle of hyphenated words. A few characters from the master file such as "┐" (upper-right corner) and the special blank are not converted into "//U" and "/I", respectively. These characters may be corrected via the MTS editor.

The page-line number ranges are specified in the form

```
ppplll¹ ppplll²
```

where "ppplll¹" is the beginning page-line number and "ppplll²" is the ending page-line number. The page-line number range

```
001001 999999
```

may be specified to convert the entire master file.

The TEXT360 source input file produced may be used as input to the TEXT360 program. However, information contained in the master file concerning page-line numbers and which lines were changed since the last +NEWLEVEL+ alter code was specified is lost.

For example, the following sequence

```
$RUN *T360MASTCONV SPUNCH=NEWINPUT PAR=MASTER=OLDMASTER
001001 004010
010001 015032
$ENDFILE
```

will convert text in the master file OLDMASTER into source input text in the file NEWINPUT. The page-line number ranges converted are from page 1, line 1 to page 4, line 10 and from page 10, line 1 to page 15, line 32.

APPENDIX C: TEXT360 CHARACTER SET

The following table lists the graphic characters of the TEXT360 character set and the keypunch or terminal characters required to represent them. The first seven entries of the table (special control characters) are the characters reserved to edit text.

Character	029 Key Stroke	Hex Rep.
	-	60
Special	+	4E
Control	*	5C
Characters	@	7C
	$	5B
	/	61
	_	6D
1	1	F1
2	2	F2
3	3	F3
4	4	F4
5	5	F5
6	6	F6
7	7	F7
8	8	F8
9	9	F9
0	0	F0
A	*A	C1
B	*B	C2
C	*C	C3
D	*D	C4
E	*E	C5
F	*F	C6
G	*G	C7
H	*H	C8
I	*I	C9
J	*J	D1
K	*K	D2
L	*L	D3
M	*M	D4
N	*N	D5
O	*O	D6
P	*P	D7
Q	*Q	D8
R	*R	D9

Character	029 Key Stroke	Hex Rep.
S	*S	E2
T	*T	E3
U	*U	E4
V	*V	E5
W	*W	E6
X	*X	E7
Y	*Y	E8
Z	*Z	E9
a	A	81
b	B	82
c	C	83
d	D	84
e	E	85
f	F	86
g	G	87
h	H	88
i	I	89
j	J	91
k	K	92
l	L	93
m	M	94
n	N	95
o	O	96
p	P	97
q	Q	98
r	R	99
s	S	A2
t	T	A3
u	U	A4
v	V	A5
w	W	A6
x	X	A7
y	Y	A8
z	Z	A9

Character	029 Key Stroke	Hex Rep.
. Period	.	4B
, Comma	,	6B
(Left Parenthesis	(	4D
) Right Parenthesis	)	5D
; Semicolon	; or /E	5E
> Greater Than	> or /F	6E
< Less Than	< or /G	4C
_ Underscore	_ or /J	6D
% Percent Sign	% or /K	6C
¢ Cent Sign	¢ or /L	4A
& Ampersand	& or /M	50
! Exclamation Point	! or /N	5A
= Equal Sign	= or /O	7E
' Apostrophe	' or /R	7D
: Colon	: or /T	7A
# Pound Sign	# or /U	7B
? Question Mark	? or /W	6F
" Quotation Mark	" or /Y	7F
¬ Logical NOT Symbol	¬ or //C	5F
| Logical OR Symbol	| or //K	4F
+ Plus Sign	/A	4E
] Right Bracket	/B	BD
▪ Square Bullet	/C	9F
$ Dollar Sign	/D	5B
± Plus or Minus	/H	9E
Special Blank	/I	09
• Bullet	/P	AF
@ Commercial "At" Sign	/Q	7C
- Minus/Hyphen	/S	60
[Left Bracket	/V	AD
* Asterisk	/X	5C
/ Slash	/Z	61
┼ Intersection	//A	8F
} Right Brace	//B	9B
≥ Greater Than or Equal To	//F	AE

Character	029 Key Stroke	Hex Rep.
≤ Less Than or Equal To	//G	8C
⌊ Lower-left Corner	//H	AB
⌋ Lower-right Corner	//J	BB
— Horizontal Line	//L	BF
≠ Not Equal Sign	//O	BE
⌉ Upper-right Corner	//U	BC
{ Left Brace	//V	8B
⌈ Upper-left Corner	//Y	AC
¤ Lozenge	//Z	9C
° Degree	//N	A1
^{1}	/1	B1
^{2}	/2	B2
^{3}	/3	B3
^{4}	/4	B4
^{5}	/5	B5
^{6} Superscripts	/6	B6
^{7}	/7	B7
^{8}	/8	B8
^{9}	/9	B9
^{0}	/0	B0
$^-$	//S	A0
$^+$	//1	8E
$^($	//2	8D
$^)$	//3	9D
¦ Broken Vertical Line[1]	//I	6A
~ Tilde[1]	//T	73
↑ Up arrow[1]	//4	74
↓ Down arrow[1]	//5	75
← Left arrow[1]	//6	76
→ Right arrow[1]	//7	77
` Grave[1]	//M	79
† Dagger[1]	//D	9A
^ Caret[1]	//E	AA
§ Section[1]	//Q	DC
¶ Paragraph[1]	//P	DD
\ Reverse slant[1]	//R	E0

[1]These characters are available only on the Xerox 9700 Page Printer using font *PGF(12).

April 1977

APPENDIX D: TEXT360 DEFAULT SPECIFICATIONS

Certain alter codes are essential for the operation of TEXT360. Should one of these codes be omitted, the default specification is assumed. The default specifications are:

```
+SINGLE+

+SWIDTH65+

+LENGTH65+

+DEPTH50+

+MARGIN0+

+HEADA+

+JUST+

+COLJUST+

+HYPHEN+

+SETTAB1=5,2=10,3=15,4=20,5=25,6=30,7=35,8=40,9=45,10=50,11=55,
12=60,13=65,14=70,15=75,16=80,17=85,18=90,19=95,20=100+

+BACKSPACEOFF+

+BLANKOFF+

+CONVERT+

+ASISOFF+

+IGNOREOFF+
```

APPENDIX E: CODING EXAMPLES

This appendix contains three coding examples. The examples show the coding required to format text, to update text, and to merge master files.

Example 1: Initial Coding

The following example shows the coding required at the outset of a document. The encoded input below produced the first three pages of the section "TEXT360" in this volume.

```
+convertoff+
+page+
+headb+
+depth53+
+swidth72+
+length72+
+backspace+
-n-
+margin0+
+ofoot+TEXT360
+efoot+TEXT360
-h1c-TEXT360
-h3-$Introduction$$
-p-TEXT360 is a text/sprocessing system that can expedite the
 production of publishable documents.
-p-The TEXT360 user specifies the desired format by embedding
 two types of instructions in the text
 flow:  "edit codes," which usually affect the format of
 a relatively small area of text, and "alter codes," which
 are more general instructions dealing with format specifications
 of a larger scope, such as page depth, column width, etc.
-p-These codes cause the text processor to "format" the
 textual material accordingly.
 The formatting capabilities include hyphenation, line
 justification, column justification, headings, indentions,
 and one/s or two/scolumn page layout.
 More complex functions include printing horizontal
 and vertical lines and placement of tabular text for
 tables and charts.
-p-The general form of an edit code is /SA/S, where the minus
 signs delimit the instruction
 and A represents the specific code desired. Some of the edit
 codes can also include numbers, but the general form
 /SA/S represents an edit code in discussions of TEXT360.
-p-The general form of an alter code is /AALTER/A, where the
 plus signs delimit the instruction and
 ALTER represents the specific alter code desired. The
 alter codes also can include numeric data, but the general
 form /AALTER/A is used in this description to represent
 alter codes.
```

-p-In the general discussion of TEXT360 procedures, the term "controls" is used when no distinction need be made between edit and alter codes.

-p-Documents produced by TEXT360 can be updated with the revision controls. The user can delete, replace, and insert text; copy text from one location to another; and replace a word or phrase by an alternative word or phrase each time the original occurs. It is also possible to create front matter listings such as a table of contents or list of figures; reformat an existing document; and merge information from two documents.

-p-The printed output of TEXT360 can be a fully formatted "camera/sready" document. The master file of the document can be created on any on/Sline storage medium.

-p-TEXT360 consists of the TEXT360 main processor program and the following peripheral programs: TEXT360 Prescan, TEXT360 Print, and TEXT360 Spelling Dictionary.

-p-The TEXT360 main program performs the following four functions:

-s1j3-/P/I/IIt encodes the text input data and updates the old master file.

-s1j3-/P/I/IIt constructs the text lines, processes the controls, and performs hyphenation and justification.

-s1j3-/P/I/IIt constructs the page formats as specified by the user.

-s1j3-/P/I/IIt creates user/sspecified lists and performs a spelling check.

-p-The TEXT360 Prescan program analyzes the input to the TEXT360 main program. It is designed to detect such errors as incorrect syntax and invalid parameters.

-p-The TEXT360 Print program prints the formatted document.

-p-The TEXT360 Spelling Dictionary program creates and maintains the dictionary required for the spelling check in TEXT360. When a spelling check is requested, TEXT360 compares each 2/s through 24/scharacter word of the document with the dictionary. Any word in the input stream that is not listed in the dictionary is printed with its location in the Spelling Check Messages section of the system/sgenerated information at the end of the document.

-p-The TEXT360 program reads the input, interprets the controls, and writes a file containing the formatted document. This file, the print file, can be printed by the TEXT360 Print program, or any other standard print program. A second file, the master file, is also created. It contains an encoded form of the document and is used for subsequent revisions. After the initial run in which the first master file is created, the user need only specify changes to the document. TEXT360 generates a print file and (optionally) a new master file with each run.

-p-The encoded input to TEXT360 is a combination of text and TEXT360 controls.

The control delimiters separate the controls from the text.
-p-Throughout this description, the controls are shown as capital letters; they may be specified using either uppercase or lowercase letters since the plus and minus sign delimiters are sufficient to distinguish control codes.
-p-Unless explicitly stated otherwise, a code must not contain blank characters.
However, there are certain alter codes of more than one word which contain blank characters to separate the words.
-p-The parameters that must accompany the controls are shown as "n" or "x",
where "n" indicates a numeric character and "x" indicates a numeric, alphabetic, or special character. The number of "n's" shown is the
same as the maximum number of decimal digits allowable. The parameter field can vary in length and, since the format of the field is not dependent
upon a specific number of digits, leading zeros can be omitted from the field.
-p-The TEXT360 character set is larger than that of the IBM 029 Card Punch and most terminal keyboards.
Those characters that cannot be entered directly from a card punch or terminal keyboard must
be specified in another way if they are to become part of the text.
-p-The uppercase and lowercase characters of TEXT360 are represented by the IBM 029
alphabetic characters and capitalization control characters.
The capitalization control characters are not explicitly needed if a terminal with both upper/s and lowercase characters is used. The remaining characters
are expressed by combining a slash with an alphabetic or numeric character. The TEXT360 character set and the corresponding punched/Scharacter configuration are listed in "Appendix C: TEXT360 Character Set."
-p-All format layouts shown in this description are governed by the following rules of notation:
-s1j3-/P/I/IBrackets /v/b indicate an optional field.
-s1j3-/P/I/Ib_/Z indicates a required blank character.
-s1j3-/P/I/IAn ellipsis (...) indicates that the preceding unit can be entered one or more times in succession.
-s1j3-/P/I/IWords composed of all uppercase letters are keywords to be supplied exactly as shown.
-s1j3-/P/I/IWords composed of all lowercase letters indicate data to be supplied by the user.
-s1j3-/P/I/IAll punctuation is literal and must be entered as shown.
-p-Both the alter codes and the edit codes are shown in an abbreviated form
when the code is discussed in general. These shortened forms should not be assumed to be necessarily valid. For the correct syntax of the alter and edit codes, refer to the appropriate discussions in the sections "Alter Codes" and "Edit Codes."

April 1977

Example 2: Updating a Master File

The following example illustrates the updating of a master file. This example is taken from the update file used to prepare Update 1 to MTS Volume 1. The original pages of MTS Volume 1 (pages 363 to 414) and the update pages may be consulted to see the output from this update file.

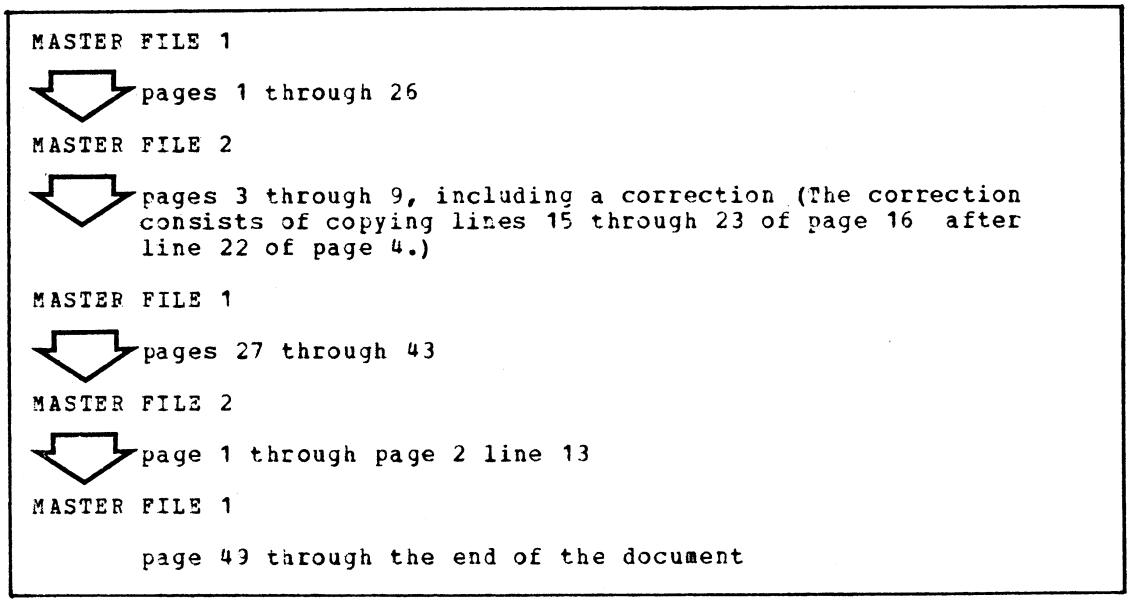
```
+convertoff+
+364036i+-p-The user may specify a local time limit on the
 /dDEBUG command, e.g.,
-s1c-/dDEBUG program T=s
-s1-The time limit specified applies only to the $execution$$
 $time$$ $of$$ $the$$ $user's$$ $program$$; time spent
 in processing SDS commands is not included.
 When the local time limit is exceeded, execution of the user's
 is suspended.  The user may subsequently
 resume execution with the CONTINUE command. When execution
 is resumed, a new local time limit is established which is the
 time limit specified by the MTS TIME option; the default is
 no time limit.
+365001rN+/
+366032,366033+ beginning with a letter is treated as a symbol;
 if it is not a defined symbol within the program, an attempt
 is made to treat it as a hexadecimal relative or absolute
 address. If the "S" component begins with a decimal digit (0/s9),
 the symbol is always treated as a relative or absolute address.
 Furthermore, an "S" component
+366038i++settab1=20,2=40+
+366040rsymbol+symbol or relative address
+366042rsymbol+symbol or relative address
+366043r-r-+
+366043i+-s0t1-XYZ-t-symbol only -r-
+367017++settab1=33+
+367025r/hj.+. Displacements may not be used on recursive
 storage reference expressions.
+369001r/a10+/a10/i/i(4 bytes)
+369002+-s1-The length is given for types E, F, P, and Z.
 Similarly,
+373031i+-p-The /qT=F and /qT=E modifiers may be applied to GRS
 and FRS, respectively, to obtain fixed/spoint and floating/spoint
 conversion of the general and floating/spoint registers.
+377013rshould+must
+377044r-s1j4-+-s0j4-/
+378027,378028+ mnemonic by one or more blanks; blanks may be
 included within the operand, but not the mnemonic. Extended
 mnemonics (such as BNM) may be used, in which case the extended
 mnemonic will include the mask digit of the BC or BCR
 instruction. The length of the instruction
+378030rthe+the first
+378032i+
-s0i9-MODIFY INS2 I'BL 0 B098'
```

```
+380025r .GT.+.GT. or >
+380026r .GE.+.GE. or >=
+380027r .LT.+.LT. or <
+380028r .LE.+.LE. or <=
+380029r .EQ.+.EQ. or =
+380030r .NE.+.NE. or ¬=
+380035r.EQ.+ = /
+381001r.EQ.+ = /
+381003r.EQ.+ = /
+381007r  .AND.+.AND. or &
+381008r  .OR.+.OR.  or |
+381009r  .XOR.+.XOR.
+381010r  .NOT.+.NOT. or ¬
+381014r.NOT.+¬/
+381014r.GT.+ > /
+381015r.EQ.+ = /
+381015r.AND.+ & /
+381015r.NE.+ ¬= /
+383007rcommmand+command
+385013rat an+or at an
+385014,385015+ by the program. For execute instructions,
 the breakpoint
+385018i+
-p-The BREAK command also may be used to set breakpoints
 in resident/ssystem storage regions. When the breakpoint is set,
 a message will be printed indicating that the breakpoint is
 a "simulator breakpoint."  In order for this type of breakpoint
 to be taken, the program must be simulated at the point of the
 breakpoint. Simulation of resident/ssystem storage may be
 achievec by specifying the SIM and
 FULLSIM options on the SET command, e.g., SIM=ON and
 FULLSIM=ON. This feature may be used for setting breakpoints
 in <EFL> (Elementary Function Library), <FIX> (FORTRAN I/zO
 Library), or PL1SYM (Resident/sSystem PL/zI Library).
 If the FULLSIM option is FULL, resident/ssystem subroutines
 in LCSYMBOL also will be simulated. Warning: Care must
 be taken in setting breakpoints in other areas of
 resident/ssystem storage since
 it cannot be guaranteed that execution of the program
 may be properly restarted.
+386018i+
-p-The SIM option may be used to specify that the program is
 to be simulated. If the SIM option is ON, any command that
 starts program execution will invoke the simulator to simulate
 the program. Resident/ssystem subroutines will not be simulated;
 instead, they will be executed normally and simulation will
 resume when the subroutine returns.
-p-The FULLSIM option may be used to specify that resident/ssystem
 subroutines are to be simulated in addition to the user program.
 If the FULLSIM option is ON, calls to <EFL> (Elementary Function
 Library), <FIX> (FORTRAN I/zO Library), and PL1SYM
 (Resident/sSystem PL/zI Library) will be simulated. If the
 FULLSIM option is FULL, calls to the resident/ssystem subroutines
```

```
 in LCSYMBOL will also be simulated. Note: The SIM option must
 also be ON for the FULLSIM option to be effective.
+398026,398027+-s1i5j5-sets a breakpoint at statement number
 27 in the program.
+401003i++repage404+
+401012i+-s1-If the TERSE option is MTS (the default), the
 current setting of the MTS TERSE option will be used as the
 setting for the SDS TERSE option.
+401027i+The WARNMSG option may be used to control certain warning
 messages. If the WARNMSG option is ON, all messages concerning
 addresses outside of control section bounds or subscripts
 outside of array bounds are suppressed.
+405010+-s0-$CL$$EAN-t2-/voption /p/p/p/b
+405028+-s0-$L$$IST -t2-/voption /p/p/p/b
+405033+-s0-$ON$$-t2-on/scondition /v;command/b
+405034i+-s0-$PARL$$IST-t2-location /vcount/b
+405038+-s0-$RESE$$T-t2-/von/scondition /p/p/p/b
+405050i+-s0-$TI$$METALLY-t2-//vON|OFF|PRINT|RESET|CLEAR//b
 /voption/p/p/p/b
+405051+-s0-$TR$$ACE-t2-trace/soption /vlocation /p/p/p/b
+406001rsection+//vsection|symbol//b
+409043,409045+ that at/spoint is executed in sequence. When
 the END command is executed, control returns to the user's
 program and normal execution is resumed (unless a breakpoint
 has been encountered during the execution of the command list).
 If control is to be
+410006rcommmand+command
+410010rat an+or at an
+410011rprogram, or at an+program.
+410012+
+410013rmacro instruction.+
+412028rat an+or at an
+412029rprogram, or at an+program.
+412030+
+412031rmacro instruction.+
+413005i+-s1-The BREAK command also may be used to set breakpoints
 in resident/ssystem storage regions. When the breakpoint is set,
 a message will be printed indicating that the breakpoint is
 a "simulator breakpoint."  In order for this type of breakpoint
 to be taken, the program must be simulated at the point of the
 breakpoint. Simulation of resident/ssystem storage may be
 achieved by specifying the SIM and
 FULLSIM options on the SET command, e.g.,  SIM=ON and
 FULLSIM=ON. This may be used for setting breakpoints
 in <EFL> (Elementary Function Library), <FIX> (FORTRAN I/zO
 Library), or PL1SYM (Resident/sSystem PL/zI Library).
 If the FULLSIM option is FULL, resident/ssystem subroutines in
 LCSYMBOL also will be simulated. Warning: Care must
 be taken in setting breakpoints in other areas of resident/ssystem
 storage since it cannot be guaranteed that execution of the
 program may be properly restarted.
```

Example 3: Merging Master Files

The following chart illustrates a hypothetical merge:



The coding required to perform the above merge (assuming all pages consist of 53 lines) is:

+026053I++INPUT+	The program begins processing master file 1 (OLDMAST) until the +INPUT+ code at the end of page 26 (+026053I+) is encountered. At this point, processing switches to master file 2 (ALTMAST) and the program rewinds master file 1.
+001001,002053+	The program bypasses pages 1 and 2 of master file 2 and starts processing at page 3.
+004022C++(016015,016023)+	The program continues to process master file 2 until line 22 of page 4 has been processed. At this point, lines 15 through 23 of page 16 are copied immediately following line 22.
+009053I++INPUT+	When the copy function is completed, processing continues until the +INPUT+ code is encountered at the end of page 9 (+009053I+) where processing switches to master file 1. The program rewinds master file 2.

+001001,026053+	The +001001,026053+ code causes the program to bypass the first 26 pages of master file 1 and to begin at page 27.
+043053I++INPUT+	Processing of master file 1 continues until the end of page 43 (+043053I+) where processing switches to master file 2. The program rewinds master file 1.
+002013I++INPUT+	At this point, processing is to begin at page 1, line 1 of master file 2; therefore, no code is entered to advance the master file. The processing of master file 2 continues until the +INPUT+ code is encountered at the end of line 13, page 2 (+002013+). Processing switches to master file 1, and the program rewinds master file 2.
+001001,048053+	The program bypasses pages 1 through 48 and starts processing at page 49. Processing continues through to the end of the master file 1.

APPENDIX F: TEXT360 CONTROLS SUMMARY

The updating alter codes are not included in this appendix. For a complete discussion of the updating alter codes, see the sections "Document Line Updating" and "Document Searching and Revising."

$ (Dollar Sign)	Starts underscoring.
$$	Ends underscoring started by $.
* (Asterisk)	Causes the letter that immediately follows it to be capitalized.
@ (At Sign)	Causes continuous capital letters.
@@	Ends the capitalization started by @.
- (Minus)	Delimits the TEXT360 Edit Codes.
+ (Plus)	Delimits the TEXT360 Alter Codes.
/ (Slash)	Specifies, when used in conjunction with an alphabetic or numeric character, a character that is not available on the keypunch or terminal keyboard or that is available on the keyboard but is assigned as a TEXT360 special control character (-, +, *, @, $, /, or _); or the end of data to be read from an input line.
_ (Underscore)	Specifies backspacing when +BACKSPACE+ is in effect.
-?nn-	Causes a user-defined, multioperation function to be performed; "nn" can be from 1 to 15.
-A-	Causes a line of text to be printed as it appears within the input flow.
-B[nnn,...,nnn]-	Begins a table (horizontal line with intersections at the column positions specified). "nnn" must be a number that lies within the specified column width.
-C-	Centers a line of text.
-D-	Begins a two-column keep.
-E-	Ends a table (horizontal line with intersections at the column positions specified in the last -E- edit code).
⁄ø-F-	Begins footnote text.

-Gnnn,nnn[,...,nnn,nnn]
: Draws a horizontal line or line segments; "nnn" must be a number that lies within the specified column width, and there must be an even number of numbers specified.

-Hn-
: Causes text to be formatted in accordance with a predefined heading format; "n" must be from 1 to 6.

-I[nn]-
: Specifies an indention of "nn" spaces; "nn" may be any value that does not reduce the line length to fewer than 20 characters.

-J[nn]-
: Specifies a hanging indention of "nn" spaces; "nn" may be any value that does not reduce the line length to fewer than 20 characters.

-K-
: Begins a regular keep.

-L-
: Draws a horizontal line within a table.

-M-
: Causes a marker to be printed beside, and a special blank to be printed within, the text line in which it is specified.

-N-
: Starts a new page.

-P-
: Starts a paragraph.

-Q[nnn,...,nnn]-
: Ends the vertical lines started by the last -V- edit code.

-R-
: Ends the section of text started by a -K-, -Y-, or -D- edit code.

-S[nn]-
: Skips "nn" lines; "nn" can be any value that does not cause the skip to exceed the page depth.

-[.]T[nn]-
: Specifies a tab position at which text is to be placed; "." causes leader periods to be printed; "nn" can be from 1 to 20.

-U-
: Indicates the end of tabular text when tabular and flowing text appear in the same line.

-Vnnn[,...,nnn]-
: Starts a vertical line at the positions specified; "nnn" must be a number that lies within the specified column width.

-W-
: Starts a new column of text.

-X[?]x-	Identifies text that can be printed in a supplemental listing; "x" can be a number from 1 to 9 or an alphabetic character.
-Y-	Begins a floating keep.
-Z-ϕ	Ends footnote text.
+ASIS+	Specifies direct text entry.
+ASISOFF+	Suppresses direct text entry.
+BACKSPACE+	Specifies backspacing.
+BACKSPACEOFF+	Suppresses backspacing.
+BLANK+	Specifies padding of input text.
+BLANKOFF+	Suppresses padding of input text.
+COLJUST+	Specifies column justification.
+COLJUSTOFF+	Suppresses column justification.
+CONVERT+	Specifies input text conversion.
+CONVERTOFF+	Suppresses input text conversion.
+DATE+x...x	Specifies the data to be printed at the top of each page; "x...x" can be up to the page width.
+DEFINE?nn=x...x+	Allows the definition of a multioperation format function; "nn" must be from 1 to 15; "x...x" can include A, C, D, H, Inn, Jnn, K, N, Snn, Xx, Y, $, and @.
+DEPTHnn+	Specifies the page depth; "nn" can be from 25 to 75.
+DOUBLE+	Prints text in a double-spaced format.
+DWIDTHnn+	Prints text in a two-column format and specifies the width of the column; "nn" can be from 20 to 52.
+EFOOT+x...x	Specifies the running foot to be printed at the bottom of even-numbered pages; "x...x" can be up to the page width.
+FRONT+	Formats all pages as right-hand pages.
+HEADx+	Specifies a heading set to be used by the program; "x" can be either "A" or "B."

+HYPHALT+word[,...,word]␣

Indicates words to be entered into the hyphenation dictionary; "word" must not be more than 24 characters, not including hyphens; the maximum number of words allowable in the dictionary is 40.

+HYPHCLR+word[,...,word]␣

Clears the old hyphenation dictionary and indicates words to be entered into the new hyphenation dictionary; "word" must not be more than 24 characters, not including hyphens; the maximum number of words allowable in the dictionary is 40.

+HYPHEN+ Specifies the hyphenation feature.

+IGNORE+ Specifies input text suppression.

+IGNOREOFF+ Suppresses input text suppression.

+INPUT[␣master]+ Allows master files to be merged.

+INITIAL+ Deletes all previous and current revision bars.

+JUST+ Specifies the line justification feature.

+LENGTHnnn+ Establishes the right margin; "nnn" can be from 20 to 107.

+LIST(x,...,x)+title

Specifies a supplemental listing to be printed; a maximum of nine lists is allowable, each with a maximum of nine values.

+MARGINnn+ Specifies the width of the left margin; "nn" can be any value up to 87, as long as +LENGTH+ minus +MARGIN+ is at least 20.

+NEWLEVEL+ Deletes all previously accumulated revision bars.

+NOHYPH+ Suppresses the hyphenation feature.

+NOJUST+ Suppresses the line justification feature.

+OFOOT+x...x Specifies the running foot to be printed at the bottom of odd-numbered pages; "x...x" can be up to the page width.

+PAGE[nnn]+ Causes new page layout from the point encountered (or page "nnn") to the end of the document; "nnn" must be an odd number of not more than three digits.

+REPAGEnnn+	Causes a new page layout from the point encountered to page "nnn"; "nnn" must be an even number of not more than three digits.
+REVISE/old+new	Allows the replacement of a word or phrase with new text each time the original is encountered in the document exactly as specified by this code; neither "old" nor "new" can exceed 31 characters, including blanks, but excluding capitalization and underscoring control characters.
+REVISEOFF/old+	Ends the text replacement started by a +REVISE+ alter code.
+SEARCH/text+	Causes a list that contains the page and line address of every occurrence of a word or phrase to be printed; "text" cannot exceed 31 characters, including blanks, but excluding capitalization and underscoring control characters.
+SEARCHOFF/text+	Ends the search started by a +SEARCH+ alter code.
+SETTAB[nn=nnn,...,nn=nnn[R]]+	Specifies the tab settings; "nn" can be from 1 to 20; "nnn" can be from 1 to 107.
+SINGLE+	Prints text in a single-spaced format.
+SPELL+	Compares each word of the input text with the words in a spelling dictionary.
+SUBTITLE+x...x	Specifies the subtitle to be printed at the top of each page; "x...x" can be up to the page width, minus the length of the date, if applicable.
+SWIDTHnnn+	Prints text in a one-column format and specifies the width of the column; "nnn" can be from 20 to 107.
+TITLE+x...x	Specifies the title to be printed at the top of each page; "x...x" can be up to the page width.

April 1977

APPENDIX G: PRESCAN ERROR MESSAGES

The TEXT360 Prescan program prints two types of messages in the system-generated information listings: error and informational. The error messages are generated when the program encounters an apparent error (see "Prescan" in the section "Appendix B: Peripheral Programs").

The informational messages serve as a reminder that the specification of a certain code requires the subsequent specification of a second code. For example, when a footnote is encountered a warning message is given that it must be ended by the -Z- edit code.

Certain error conditions can cause error messages to be issued for valid input. These error conditions cause the scan to be out of synchronization; consequently, some input will be processed before the program can recover.

The number of errors permitted is specified in the PAR field by the MAJ=mm and MIN=nn parameters.

If the PAR field is omitted, the condition MAJ=0 and MIN=5 is assumed. The informational or warning messages are not counted as either major or minor errors.

The Prescan program returns with a return code of 8 if the number of allowable major errors is exceeded. If the number of major errors is not exceeded, the program returns with a return code of 4 if the number of allowable minor errors is exceeded. Otherwise, the program returns with a return code of 0.

The list of error messages that follows includes a brief explanation of each message and whether it is a major or a minor error. The explanation does not include the corrective measure. This must be obtained by referring to the applicable TEXT360 control to determine the valid conditions.

+ DELIMITER MISSING;

Explanation: An alter code delimiter (+) is not specified. (Major error)

ALTER CODE;

Explanation: An invalid alter code is specified. (Major error)

BAD ?N;

Explanation: The -?- code specifies a value greater than 15. (Major error)

B,V,G CODE;

Explanation: The -B-, -V-, or -G- code specifies a column position greater than 107, or the -B- code specifies only one column position, or the number of column positions specified in the -G- code is not a multiple of two. (Minor error)

BB OR VV CODE;

Explanation: The -BB- or -VV- code is specified without column positions. (Minor error)

CHAR REPLACE;

Explanation: An invalid character is specified within the "old" portion of the +pppllRold+new code. (Minor error)

DEFINE;

Explanation: A +DEFINE+ code is specified incorrectly. (Major error)

DEPTH;

Explanation: The +DEPTH+ code specifies a page depth of fewer than 25 or more than 75 lines. (Minor error)

DWIDTH;

Explanation: The +DWIDTH+ code specifies a column width of fewer than 20 or more than 52 characters. (Minor error)

EDIT CODE;

Explanation: An invalid edit code is specified. (Major error)

HCODE;

Explanation: The -Hn- code is specified as -H-, -H0-, or -Hn- where "n" is greater than 6. (Minor error)

I,J CODE;

Explanation: The -I- or -J- code specifies an indention or hanging indention greater than 87. (Minor error)

ILLEGAL ED-DEFINE;

Explanation: The +DEFINE+ code contains an unacceptable edit code. (Major error)

INPUT RECORD TOO LONG;

Explanation: An input record is longer than 256 characters. (Major error)

INV CHAR;

Explanation: A character not included in the TEXT360 character set is specified. (Minor error)

INV UPDATE CODE;

Explanation: The update code is specified incorrectly. (Major error)

INV WORD HYP;

Explanation: The word to be added to the hyphenation dictionary contains either fewer than 2 or more than 24 characters. (Minor error)

INVALID COPY;

Explanation: The second page and line number enclosed by the parentheses in the +ppplllC(ppplll,ppplll)+ code is equal to or less than the first page and line number. (Major error)

LENGTH;

Explanation: The +LENGTH+ code specifies a line length of fewer than 20 or more than 107 characters. (Minor error)

LIST;

Explanation: The syntax of the +LIST+ code is incorrect, or the (x,...,x) portion of the +LIST+ code contains a character other than a numeric or alphabetic character. (Major error)

MARGIN;

Explanation: The +MARGIN+ code specifies a margin of more than 87 characters. (Minor error)

NO BLANK FCODE;

Explanation: The -F- code is not preceded by a blank character. (Minor error)

NO EDIT CODE DELIM;

Explanation: An edit code delimiter (-) is not specified. (Major error)

PAGE NO;

Explanation: The +PAGE+ or +REPAGE+ code specifies a page number greater than 999, or the +REPAGE+ code does not specify a page number. (Major error)

QCODE;

Explanation: The -Q- code specifies a column position of more than 107 characters. (Minor error)

RCODE IN COMB;

Explanation: The -R- code is not the last edit code of the combination in which it is specified. (Major error)

RELEASE;

Explanation: An -R- or -Z- code is specified. This message is given to provide easy recognition of the -R- or -Z- code. (Informational message)

REPLACE TOO LONG;

Explanation: The "old" portion of the +ppplllRold+new code specifies more than 116 characters. (Minor error)

S,R TOO LONG;

Explanation: The search argument of the +SEARCH+ ("text") or +REVISE+ ("old") code exceeds 31 characters. (Major error)

SCODE;

Explanation: The -S- code specifies a skip of more than 75 lines. (Minor error)

SPEC CHAR;

Explanation: The search argument of the +SEARCH+ ("text") or +REVISE+ ("old") code contains an invalid character. (Major error)

Note: This error message is issued if any character other than a number or alphabetic character is included in the +SEARCH+ or +REVISE+ code, even though it may be valid.

SWIDTH;

Explanation: The +SWIDTH+ code specifies a column width of fewer than 20 or more than 107 characters. (Minor error)

TABS;

Explanation: The syntax of the SETTAB code is incorrect, or the +SETTAB+ code defines a tab number greater than 20. (Major error)

TCODE;

Explanation: The -T- code specifies a tab number greater than 20. (Major error)

TOO MANY ED CODE IN COMB;

Explanation: More than six edit codes have been specified within one pair of delimiters. (Major error)

TOO MANY S,R REQ;

Explanation: More than 64 +SEARCH+ and +REVISE+ codes have been specified within the input. (Minor error)

TOO MANY WDS HYP;

Explanation: The number of words specified in the +HYPHALT+ or +HYPHCLR+ code causes the hyphenation dictionary to exceed 40 words. (Major error)

UPDATE SEQ INC;

Explanation: The page and line numbers of the updating codes are not in ascending order. (Major error)

WARNING -D- CODE;

Explanation: A -D- code is specified. The two-column keep requires a subsequent release (-R-). (Informational message)

WARNING -F- CODES;

Explanation: An -F- code is specified. The footnote requires a subsequent release (-Z-). (Informational message)

WARNING -K- CODE;

Explanation: A -K- code is specified. The regular keep requires a subsequent release (-R-). (Informational message)

WARNING -Y- CODE;

Explanation: A -Y- code is specified. The floating keep requires a subsequent release (-R-). (Informational message)

XCODE;

Explanation: The -X- code specifies a character other than a number or alphabetic character as a parameter. (Major error)

Z IN COMB;

Explanation: The -Z- code is specified in combination with other edit codes. (Major error)

April 1977

APPENDIX H: TEXT360 ERROR MESSAGES

TEXT360 can detect certain error conditions within the input text. If an error is detected, the program issues the appropriate error message in the error message listing. The program lists the error message number, the error message and, in the line immediately following the error message, the page and line number at which the error occurred and the erroneous text. Messages T360.001 through T360.015 are printed with the input line and MTS file line numbers, and if applicable, the master file page and line numbers.

Each error message listed below is accompanied by a brief explanation of the error message and the action the program takes. The program action affects only the print file; the erroneous data, as specified in the input text, is written on the master file and must be corrected by the user.

When an updating error is encountered, any new data in the updating code is inserted at the location the error is encountered. For example, if a +REVISE+ code specifies incorrectly the "old" text and the program cannot locate the text on the specified line as a result, the "new" text is inserted at the end of that line.

Hyphens are included in the error messages listed below for the sake of clarity. When printed in the error message listing, the hyphens do not appear in the message.

T360.001 -- NO DELIMITER FOLLOWING INSERT CODE

Explanation: The terminal delimiter (+) is missing from the +pppllllI+ code.

Program Action: The program ignores the insert code.

T360.002 -- UPDATE CODE OUT OF SEQUENCE

Explanation: The page and line number of the code is less than that of the one that precedes it.

Program Action: The program ignores the code that is out of sequence; however, any new data in the erroneous code is inserted following the line number of the code that precedes it.

T360.003 -- WORD REPLACE EXCEEDS 116 CHARACTERS

Explanation: The "old" portion of the +ppplllRold+new code exceeds 116 characters.

Program Action: The program ignores the complete replace code.

T360.004 -- WORD REPLACE NOT FOUND AT SPECIFIED LOCATION

Explanation: The "old" portion of the +ppplllRold+new code cannot be located at the page and line specified.

Program Action: The program inserts the new data following the line number specified.

T360.005 -- COPY SPECIFIED AS AN UPDATE TO COPIED TEXT

Explanation: A request to copy text into text that has been copied is specified. This type of copy must be specified as three separate copies.

T360.006 -- INVALID COPY CODE OR COPYOFF SYNTAX

Explanation: The +ppplllC+ or +CPYOFF+ code is specified incorrectly.

Program Action: The program ignores the code.

T360.007 -- UPDATE CODE HAS IMPROPER DELIMITER

Explanation: The update code is specified incorrectly.

Program Action: The program ignores the update code.

T360.008 -- SPECIFIED PAGE-LINE NUMBER NOT ON OLD MASTER FILE

Explanation: The page and line number specified for the update code is not on the master file.

Program Action: The program ignores the update code; however, any new data in the erroneous code is inserted at the location at which the program determines the erroneous page-line number does not exist on the old master file.

T360.009 -- INVALID ENCODED CHARACTER

Explanation: The input contains a character not included in the TEXT360 character set.

Program Action: The program ignores the invalid character.

T360.010 -- TEXT360 CANNOT REPOSITION TO START OF COPY

Explanation: The TEXT360 program cannot reposition the old master file to the location of processing before the copy was initiated.

Program Action: The program continues to process from the copied location.

T360.011 -- ALTER CODE PROCESSING OUT OF SYNCHRONIZATION

Explanation: The program, as the result of improper input, is processing text as codes and codes as text.

Program Action: The program can determine after an indeterminate amount of text has been processed that it is out of synchronization and corrects itself.

T360.012 -- REMAINING INPUT NOT PROCESSED BECAUSE OF ERRONEOUS UPDATE SPECIFICATION

Explanation: The update to the master file is specified in such a way that the program cannot process it.

Program Action: The program prints the first line of the flushed input immediately following the error message.

T360.013 -- TOO MANY SEARCH AND/OR REVISE REQUESTS; REQUESTS IGNORED

Explanation: When the +SEARCH+ or +REVISE+ code was encountered, 64 such codes were operative.

Program Action: The program ignores the code.

T360.014 -- INVALID SEARCH OR REVISE FORMAT

Explanation: The +SEARCH+ or +REVISE+ code is either specified incorrectly or the character count exceeds 31.

Program Action: The program ignores the code.

T360.015 -- SEARCHOFF OR REVISEOFF REQUEST NOT FOUND

Explanation: The search argument of the +SEARCHOFF+ or +REVISEOFF+ code is not specified exactly the same as in the code it is to end.

Program Action: The program ignores the code.

T360.101 -- INVALID n VALUE SPECIFIED FOR -Hn- CODE; -I0- ASSUMED

Explanation: The -H- code specifies a value of zero or greater than 6.

Program Action: The invalid -H- code is accepted by the program; however, it is processed as an -I0- code.

T360.102 -- n NOT SPECIFIED FOR -Hn- CODE; -I0- ASSUMED

Explanation: The -H- edit code without a specified value is an invalid code. The code must be specified with a number from 1 to 6.

Program Action: The invalid -H- code is accepted by the program; however, it is processed as an -I0- code.

T360.103 -- n NOT SPECIFIED FOR -?n- CODE; -I0- ASSUMED

Explanation: The -?- code without a specified value is an invalid code. The code must be specified with a number from 1 to 15.

Program Action: The invalid -?- code is accepted by the program; however, it is processed as -I0- code.

T360.104 -- INVALID n SPECIFIED FOR -?n- CODE; -I0- ASSUMED

Explanation: The -?- code specifies a value of zero or greater than 15.

Program Action: The invalid -?- code is accepted by the program; however, it is processed as -I0- code.

T360.105 -- x NOT SPECIFIED FOR -Xx- CODE; NO EXCEPTION GENERATED

Explanation: The -X- edit code is specified without a value.

Program Action: The invalid -X- code is accepted by the program; however, the program does not "tag" the text line with an exception value.

T360.106 -- -Snn- EXCEEDS PAGE DEPTH; CODE SET TO -S1-

Explanation: The skip specified by the -S- code causes the page depth to be exceeded.

Program Action: The program sets the -S- code to -S1-.

T360.107 -- n VALUE GREATER THAN 20 SPECIFIED IN -Tn- CODE; CODE SET TO -T1-

Explanation: The -T- edit code specifies a value greater than 20.

Program Action: The program sets the -T- code to -T1-.

T360.120 -- INVALID ALTER CODE

Explanation: The alphabetic portion of the alter code does not represent a valid code.

Program Action: The alter code is accepted by the program; however, the only action initiated by the code is to start a new line. The code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.121 -- NO NUMERIC VALUE FOLLOWING ALTER CODE; MAXIMUM ASSUMED

Explanation: The alter code is specified without a numeric value.

Program Action: The program assumes the maximum parameter of the code.

T360.123 -- COLUMN WIDTH GREATER THAN MAXIMUM; SET TO MAXIMUM

Explanation: The number of characters specified for the column width exceeds the maximum number that can be specified.

Program Action: The program sets the column width equal to the maximum number of characters that can be specified.

T360.124 -- COLUMN WIDTH LESS THAN MINIMUM; SET TO 20

Explanation: The number of characters specified for the column width is less than 20.

Program Action: The program sets the column width to 20.

T360.125 -- MARGIN EXCEEDS MAXIMUM; SET TO MAXIMUM

Explanation: The value specified by the +MARGIN+ code exceeds the maximum that can be specified.

Program Action: The program sets the margin to the maximum number of characters that can be specified.

T360.126 -- LINE LENGTH GREATER THAN COLUMN WIDTH; SET TO COLUMN WIDTH

Explanation: The number of characters specified for the line length exceeds the number of characters specified for the column width.

Program Action: The program sets the line length equal to the column width.

T360.127 -- LINE LENGTH LESS THAN MINIMUM; SET TO 20

Explanation: The number of characters specified for the line length is less than 20.

Program Action: The program sets the line length to 20.

T360.128 -- PAGE NUMBER SPECIFIED AFTER PAGE PROCESSED; CODE IGNORED

Explanation: The +PAGE+ or +REPAGE+ code is entered after the page specified by the code has been processed.

Program Action: The program accepts the code, but it is not processed.

T360.129 -- CLOSING DELIMITER NOT SPECIFIED IN ALTER CODE, DELIMITER ASSUMED

Explanation: The alter code is entered with an opening delimiter, but the closing delimiter is not specified.

Program Action: The program assumes the presence of a closing delimiter.

T360.130 -- n NOT SPECIFIED FOR J CODE WITHIN +DEFINE; J IGNORED

Explanation: The -J- code within the +DEFINE+ code does not specify a value.

Program Action: The erroneous portion of the +DEFINE+ code is ignored by the program; the remainder of the code is processed.

T360.131 -- HANGING INDENTION VALUE WITHIN +DEFINE CODE TOO LARGE

Explanation: The line length minus the margin and hanging indention is less than 20.

Program Action: The program assumes a hanging indention of zero.

T360.132 -- INDENTION VALUE WITHIN +DEFINE CODE TOO LARGE

Explanation: The line length minus the margin and indention is less than 20.

Program Action: The program assumes an indention value of zero.

T360.133 -- n NOT SPECIFIED FOR I CODE WITHIN +DEFINE; I IGNORED

Explanation: The -I- code within the +DEFINE+ code does not specify a value.

Program Action: The erroneous portion (-I-) of the +DEFINE+ code is ignored by the program; the remainder of the code is processed.

T360.134 -- n NOT SPECIFIED FOR S CODE WITHIN +DEFINE; S IGNORED

Explanation: The -S- code within the +DEFINE+ code does not specify a value.

Program Action: The erroneous portion (-S-) of the +DEFINE+ code is ignored by the program; the remainder of the code is processed.

T360.135 -- INVALID ENTRY WITHIN +DEFINE CODE; INVALID ENTRY IGNORED

Explanation: An unacceptable code is specified within the +DEFINE+ code.

Program Action: The invalid portion of the +DEFINE+ code is ignored by the program; the remainder of the code is processed.

T360.136 -- = ASSUMED IN +DEFINE CODE

Explanation: The equal sign is not specified with the +DEFINE+ code.

Program Action: The program assumes the presence of an equal sign.

T360.137 -- ? NOT FOUND IN +DEFINE CODE

Explanation: The question mark is not specified with the +DEFINE+ code.

Program Action: The program ignores the code.

T360.138 -- NO NUMERIC VALUE FOLLOWING ? IN +DEFINE CODE

Explanation: The numeric value that must follow the question mark in the +DEFINE+ code is not specified.

Program Action: The program accepts the code; however, it does not initiate any action; the code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.139 -- INVALID NUMBER FOLLOWING ? IN +DEFINE CODE

Explanation: The +DEFINE+ code specifies a value of zero or greater than 15.

Program Action: The program ignores the code.

T360.140 -- INVALID TAB NUMBER SPECIFIED IN +SETTAB CODE

Explanation: A tab number (indicated by "nn" in the +SETTAB+ code) is not specified or specifies a nonnumeric character.

Program Action: The program accepts and processes the code up to the point of the error; the remainder of the code is ignored; the complete code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.141 -- INVALID COLUMN POSITION SPECIFIED IN +SETTAB CODE

Explanation: A column position (indicated by "nnn" in the +SETTAB+ code) is not specified or specifies a nonnumeric character.

Program Action: The program accepts and processes the code up to the point of the error; the remainder of the code is ignored; the complete code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.142 -- TAB NUMBER WITHIN +SETTAB GREATER THAN 20

Explanation: The +SETTAB+ code specifies a tab number greater than 20.

Program Action: The erroneous tab number and its column position are ignored by the program; the remainder of the code is processed; the complete code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.143 -- COLUMN POSITION WITHIN +SETTAB GREATER THAN 107

Explanation: The +SETTAB+ code specifies a column position greater than 107.

Program Action: The erroneous column position and its tab number are ignored by the program; the remainder of the code is processed; the complete code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.144 -- PAGE DEPTH GREATER THAN MAXIMUM; SET TO 75

Explanation: The number of lines specified for the page depth exceeds 75.

Program Action: The program sets the page depth to 75.

T360.145 -- PAGE DEPTH LESS THAN MINIMUM; SET TO 25

Explanation: The number of lines specified for the page depth is less than 25.

Program Action: The program sets the page depth to 25.

T360.146 -- HYPHENATION DICTIONARY OVERFLOWED

Explanation: The number of words entered in the hyphenation dictionary exceeds 40.

Program Action: The first 40 words submitted to the dictionary are entered; the remaining words are ignored.

T360.147 -- WORD IN HYPHENATION DICTIONARY TOO LONG; DELETED

Explanation: A word entered in the hyphenation dictionary is more than 24 characters long.

Program Action: The word is deleted.

T360.148 -- LINE OVERFLOW; DATA LOST

Explanation: Data, such as a 120-character word, could not be entered during the hyphenation process.

Program Action: The data is ignored.

T360.151 -- INSUFFICIENT INFORMATION FOLLOWING -G- CODE

Explanation: The -G- code is specified with only one column position.

Program Action: The -G- code is accepted by the program; however, no line is drawn; the code is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.152 -- NO COLUMN POSITIONS SPECIFIED WITH -G- CODE

Explanation: The line parameter is not specified with the -G- code. (-G- with no numbers does not print the last specified horizontal line.)

Program Action: The -G- code is accepted by the program; however, no processing takes place; it is assigned a page and line number and is printed in the alter codes, expanded edit codes, and footnotes listing.

T360.153 -- ODD NUMBER OF COLUMNS SPECIFIED WITH -G- CODE

Explanation: The number of column positions specified by the -G- code is an odd number (not a multiple of two).

Program Action: The complete code is accepted by the program; however, the last unpaired column position is not processed.

T360.154 -- -Qnn- CODE FOR A NONEXISTENT VERTICAL LINE

Explanation: The -Q- code specifies the ending of a nonexisting vertical line.

Program Action: If the code specifies the ending of only one vertical line, the code is accepted by the program, but it is not processed; if the code specifies the ending of several lines, the code is processed.

T360.155 -- NO COLUMN POSITION SPECIFIED WITH -BB- CODE

Explanation: The -BB- code is specified without column positions.

Program Action: The code is accepted by the program, but it is not processed.

T360.156 -- NO COLUMN POSITION SPECIFIED WITH -VV- CODE

Explanation: The -VV- code is specified without column positions.

Program Action: The code is accepted by the program, but it is not processed.

T360.157 -- INSUFFICIENT INFORMATION FOLLOWING -B- CODE

Explanation: The -B- code is specified with only one column position.

Program Action: The code is accepted by the program, but it is not processed.

T360.158 -- NUMBER OF VERTICAL LINES STARTED BY -B- AND/OR -BB- CODES EXCEEDS 22

Explanation: The total number of vertical lines started by the -B- or -BB- code exceeds 22.

Program Action: The program processes the first 22 lines requested, but the remaining requests are ignored by the program.

T360.160 -- ATTEMPT TO OVERLAY VALID CHARACTER WHILE CONSTRUCTING A TABLE

Explanation: The -B- or -E- edit code specifies a column position that already contains text.

Program Action: The program suppresses the line character and prints the text.

T360.161 -- ATTEMPT TO PLACE VERTICAL LINE BEYOND COLUMN WIDTH

Explanation: The -V- code specifies a column position greater than the column width.

Program Action: The program accepts the code, but it is not processed.

T360.162 -- ATTEMPT TO OVERLAY VALID CHARACTER WHILE DRAWING HORIZONTAL LINES

Explanation: The -G- code specifies a horizontal line to be printed in a line position that already contains text.

Program Action: The program suppresses the horizontal line character and prints the text.

T360.163 -- BEGIN TABLE LINE DRAWN BEYOND COLUMN WIDTH

Explanation: The -B- code specifies a column position greater than the column width.

Program Action: The program accepts the complete code; however, the table line is drawn to the column width only.

T360.164 -- ATTEMPT TO END NONEXISTENT TABLE

Explanation: An -E- code is specified while no -B- code is operative.

Program Action: The program accepts the code, but it is not processed.

T360.165 -- TOO MANY VERTICAL LINES SPECIFIED

Explanation: The total number of vertical lines specified by the -V- or -VV- code exceeds 22.

Program Action: The program processes the first 22 lines requested, but the remaining lines are ignored.

T360.167 -- ATTEMPT TO OVERLAY VALID CHARACTER WHILE DRAWING VERTICAL LINE

Explanation: The -V- code specifies a column position that already contains text.

Program Action: The program suppresses the vertical line character and prints the text.

T360.168 -- TAB FIELD OVERLAID

Explanation: The -T- code specifies a tab position that already contains text.

Program Action: The program suppresses the original text and prints the tabbed text.

T360.181 -- AS-IS TEXT EXCEEDS 140 CHARACTERS

Explanation: The length of the text that is under the control of one -A- code exceeds 140 characters.

Program Action: The program starts a new line with the extra characters.

T360.182 -- INVALID EDIT CODE

Explanation: The alphabetic portion of the edit code does not represent a valid edit code.

Program Action: The edit code is accepted by the program, but the invalid portion of the code is not processed.

T360.183 -- EDIT CODE PROCESSING OUT OF SYNCHRONIZATION

Explanation: The program, as the result of improper input, is processing text as edit codes and edit codes as text.

Program Action: The program can determine after an indeterminate amount of text has been processed that it is out of synchronization and it corrects itself.

T360.184 -- AS-IS TEXT EXCEEDS COLUMN WIDTH

Explanation: The number of printable characters of the text line exceeds the column width.

Program Action: The program truncates (on the print file only) the text at the column width.

T360.186 -- LINE LENGTH MINUS MARGIN AND INDENTION IS LESS THAN 20

Explanation: The line length minus the margin and indention reduces the printable line to fewer than 20 characters.

Program Action: The program sets the indention to the maximum indention that permits a 20-character line.

T360.187 -- LINE LENGTH MINUS MARGIN AND HANGING INDENTION IS LESS THAN 20

Explanation: The line length minus the margin and hanging indention reduces the printable line to fewer than 20 characters.

Program Action: The program sets the hanging indention to the maximum indention that permits a 20-character line.

T360.188 -- MORE THAN 6 EDIT CODES IN COMBINATION

Explanation: More than six edit codes in one combination are specified. (This message usually indicates that processing is out of synchronization.)

Program Action: The first six codes are processed; however, starting at the seventh code and continuing to the next edit code, the input is processed as text.

T360.189 -- KEEP EXCEEDS PAGE DEPTH; KEEP RELEASED

Explanation: The space required for the specified regular or floating keep is greater than that of one column.

Program Action: The program releases the keep code at the point that the keep exceeds the length of the column.

T360.190 -- FOOTNOTE MORE THAN 20 LINES; FOOTNOTE STOPPED

Explanation: A footnote contains more than 20 lines.

Program Action: Only 20 lines are retained as a footnote. The remainder of the footnote is entered as unfootnoted text on the following page.

T360.201 -- LINE COUNT OVER 999; REPAGE MODE FORCED

Explanation: A page and its supplementary pages have more than 999 lines.

Program Action: The current supplementary page is terminated at line 999 and the document is forced out of logical sheet mode.

T360.202 -- MORE THAN 999 PAGES; PAGE SET TO 0

Explanation: A document has more than 999 pages.

Program Action: The page counter is reset to 0.

APPENDIX I: EXTENSIONS TO TEXT360

This appendix lists the extensions that have been added to the IBM version of TEXT360 to produce the MTS version. The IBM version was described in the IBM publication, IBM System/360 TEXT360 Introduction and Reference Manual, form C35-0002, which is now out of print.

- The TEXT360 program has been converted from a PL/I program into a 360/370-assembly language program. The four phases in the IBM version, FMAINT, BLDLINE, PLAOUT, and POSTPR, have been combined into one program. The use of internal files, except possibly for EXCEPTN, has been eliminated. The language conversion and phase combination reduces the execution time by a factor of three.

- TEXT360 has been converted to a terminal as well as card-oriented program.

- The global hyphenation dictionary feature has been added (see the description of the HYPHDICT parameter in "Appendix B: Peripheral Programs").

- The length of entries in the local hyphenation dictionary (+HYPHALT+) has been extended from 20 to 24 characters. The local dictionary is searched in reverse order of entry so that the latest added entries are used first.

- The length of revision that may be specified by the +pppl1lR+ alter code has been extended from 20 to 116 characters.

- The length of entries for the +SEARCH+ and +REVISE+ alter codes has been extended from 20 to 31 characters. The number of entries that can be active has been extended from 60 to 64.

- The number of lines allowed in a footnote has been extended from 10 lines to 20 lines.

- The +INPUT+ alter code has been changed to allow the merging of more than two master files.

- The +ASIS+ and +ASISOFF+ alter codes have been added to allow direct (untranslated) text entry.

- The +BACKSPACE+ and +BACKSPACEOFF+ alter codes have been added to allow backspacing (overprinting).

- The +BLANK+ and +BLANKOFF+ alter codes have been added to allow the padding of the input with blanks.

- The +CONVERT+ and +CONVERTOFF+ alter codes have been added to allow the direct input of upper- and lowercase text.

- The +IGNORE+ and +IGNOREOFF+ alter codes have been added to allow input text suppression.

- The +INITIAL+ alter code has been added to delete current revision bars.

- The column width, line length, and page depth defaults have been changed from +DWIDTH43+, +LENGTH43+, and +DEPTH68+ to +SWIDTH65+, +LENGTH65+, and +DEPTH50+, respectively.

- The behavior of the -B- edit code has been changed so that a second -B- code terminates the table drawn by the first -B- code with a horizontal line. With the IBM version, the first box is not terminated with a horizontal line.

- The behavior of the -G- and -V- edit codes has been changed so that vertical and horizontal lines may be specified together in either order. With the IBM version, the horizontal lines must be specified first.

- The -Xn- edit code has been expanded to the form -X?n- to allow information to be included in the supplemental listings without being included in the document.

- TEXT360 error messages are separated from the listing of alter codes, expanded edit codes, and footnotes. Many error messages have been expanded to give more useful information.

- The length of the entry for the +DATE+ alter code has been extended from 18 characters to the page width.

Page Revised February 1979 April 1977

MISCELLANEOUS TEXT-PROCESSING PROGRAMS

| The public file descriptions are taken from MTS Volume 2, Public File | Descriptions. These are programs which may be of use in text processing.

Page Revised February 1979 April 1977

*DITTO

Contents: The object module of the ditto-master preparation program.

| Purpose: To convert output from FORMAT or TEXT360 into a format
| suitable for preparing a ditto master.

Use: The program is invoked by the $RUN command.

| Program Key: *DITTO

Logical I/O Units Referenced:
SCARDS - text input (as produced by FORMAT or TEXT360).
SPRINT - reformatted output.
SERCOM - error diagnostics and prompting messages.
GUSER - responses to prompting messages.

| Description: This program produces output suitable for any terminal
| with an operational backspace character; this includes
| most terminals except for the Teletype terminals. Typical applications of this program are to produce ditto masters by inserting the masters into the terminal
| carriage and having the output typed directly onto the
| master, and to produce output on bond paper.

If SPRINT is assigned to a device type other than any of
| the above, overprinted characters will be printed on the
| line below the characters they are supposed to overstrike.

Before invoking *DITTO, the user should issue the following device control commands (even if the terminal has a carriage width less than 255):

```
LEN=255
RM=255
```

These device commands are necessary because the output may consist of lines containing backspace characters. The number of characters in the line may be greater than the carriage width, but the text will fill the proper number of columns on the page.

When the program is invoked, it will initially print the message:

```
TYPE "RETURN" WHEN READY FOR TYPED COPY
```

At this point, the ditto master or other sheet of paper should be placed in the terminal carriage, lined up one line above where the first line is to start. When this is done, the RETURN key should be depressed. The first page of the document will be typed; when this page is finished, the program will pause. At this point, a new master may be inserted and the entire procedure repeated. This may be continued for as many pages as desired.

The program will interpret the first character of each input line as a carriage-control character in the same manner as for line printers. Lines with carriage-control characters that would cause overprinting (e.g., for underlining) will be converted into a sequence of print characters and backspace characters which will produce the same result. Pages are identified by the "1" logical carriage-control character (as produced by FORMAT) or by the X'8B' machine carriage-control character (as produced by TEXT360).

If any difficulty arises in the middle of a page, an attention interrupt may be given which will interrupt the printing of that page. The program will rewind the input file to the beginning of the current page. This page may then be repeated after inserting a new sheet.

The program will terminate after the last page in the input file is printed or when the user enters an end-of-file instead of entering RETURN for a new page to begin.

The program can also be used to selectively scan for a particular page in the document. A page scan is requested by typing an integer page number before entering RETURN. This number is used to count the pages from the beginning of the input file. Note that this may not necessarily correspond to any document page; e.g., entering 5 will cause the fifth page from the beginning to be printed whether or not it is document page number 5. When using this feature, it is advisable to first insert a sheet of scrap paper into the terminal, type the desired page number, and then RETURN. The page will begin to print and the user can verify that it is the desired page. The page may then be interrupted, the ditto master inserted, and then restarted at the beginning. After typing a page in this manner, the program will print the following page if another RETURN is entered.

*INDEX

Contents: The object module of the index-generating program.

Purpose: To generate indices for books, articles, collections of slides, or any other data collection.

Use: This program is invoked by the $RUN command.

| Program Key: *INDEX

Logical I/O Units Referenced:

SCARDS - the input data to form the index.
SPUNCH - the final output; either in the form of input to FORMAT, input to TEXT360, or printer output to be $COPYed to a printer.

Parameters: The following parameters may be specified in the PAR field of the $RUN command. The parameter list must begin with a semicolon ";" so that the parameter list will be passed to the program. The underlined portion of each parameter may be used as an abbreviation.

WIDTH=n The column width for the text of index (defaults to 35).

LINES=n The number of lines per page for the index (defaults to 53).

FORMAT={FORMAT|TEXT360|OUTPUT}
The type of text output to be produced (defaults to OUTPUT - printer output).

| COLUMNS=n The number of columns on a page (1-6 for
| FORMAT, 1 or 2 for TEXT360, or 1 for
| OUTPUT). The default is 1 column.

ENTRIES=n The maximum number of entries to be entered into the index (defaults to 1000).

Description: The input for the index-generating program should be organized in the form of a line file or card deck. Each line contains a location attribute followed by one or more entry attributes. The location attribute (starting in column 1) is the specific page number, range of pages, slide number, etc., giving the location of the item in the text being indexed. This may be any character string and is not restricted to page numbers. The location attribute is terminated by a blank (if column 1 is blank, the location will be the same as the previous item). The

subsequent fields in the line are separated by one or more blanks. These fields describe the terms in the text which are to become the index headings and subheadings. Each entry attribute consists of a heading followed optionally by one or more subheadings. All entry attributes on each line are associated with the location attribute for that line. Each subheading is indicated by a colon ":" and is appended to the entry attribute (with no intervening blanks). A maximum of four levels of subheadings are allowed; the first subheading forms the first level, the second subheading forms the second level, etc. When the final index is generated, all of the subheadings will be alphabetized and indented under their associated headings. Each level of subheading will be indented three spaces. If an entry attribute with no location attribute is desired, the entry should be entered as

```
# entry
```

with the pound sign in column 1.

All input may be entered in upper- and lowercase letters. If lowercase letters are desired, the input must be entered via a terminal capable of transmitting lowercase letters; uppercase conversion must be turned off via the UC=OFF device command for the Data Concentrator or the Memorex 1270, or the %LC device command for 3270-type terminals.

If a blank is to be included as part of an entry or location attribute, a pound sign "#" must be inserted into the entry in place of the blank, e.g.,

```
4 Data#Concentrator
```

A blank will replace the pound sign in the final output. If a pound sign is to be inserted in the entry, it must be preceded by an exclamation mark "!". If a colon is to be included in an entry, two colons must be used, e.g.,

```
11 MTS::#Michigan#Terminal#System
```

will appear as

```
MTS:Michigan Terminal System, 11
```

Quotation marks are ignored in the alphabetization of entries, e.g.,

```
5 "An#article"
```

is alphabetized under "A" rather than grouped with quotation marks. All special characters, other than the double quote, are ordered at the front of the index. Location attributes containing groups of digits separated by periods or other punctuation are ordered by treating each group of digits as a separate integer; e.g., the following location attributes are ordered as shown:

```
25.2.5  25.2.14  25.10
```

Except for capitalization, alphabetization follows the 360-collating sequence.

The following example illustrates the use of headings and subheadings in preparing an index. The input

```
1 Files:changing
2 Files:changing:sequential
3 Files:line
4 Full#duplex:operation
```

will generate the index

```
FILES,
   CHANGING, 1
      SEQUENTIAL, 2
   LINE, 3
FULL DUPLEX,
   OPERATION, 4
```

It is assumed that uppercase conversion was not turned off.

A provision is made to allow the indexing of text with chapters, or other divisible units, in segments. Each segment is indexed relative to the beginning of that segment rather than to the entire text. In this way, a chapter can be indexed before the final location in the main text is known. The $FIRST and $BASE commands may be used for this purpose. The $FIRST command specifies the base of the final copy used to print the document; this command is given in the form

$FIRST PAGE OF CHAPTER=n

where "n" is the final page number of the first page of the segment. The $BASE command specifies the base of the proof copy used to prepare the index; this command is given in the form

$BASE OF CHAPTER=n

where "n" is the current page number for the first page of the segment. For example, if the index is prepared from a proof copy in which the first page of the chapter begins on page 19, the command $BASE=19 should be specified in the input to *INDEX. When the entire document is ready for final printing, the final page number of the first page of the chapter should be specified in addition to the proof base in the input to *INDEX, e.g.,

```
$FIRST=23
$BASE=19
```

When a group of digits is followed immediately by one or more groups consisting of a period followed by digits, only the first group of digits is affected by the $FIRST and $BASE commands; e.g.,

```
$FIRST=151
27.8.36-30 Operations
```

would result in

```
Operations, 177.8.36-180
```

The FORMAT parameter may be used to specify the type of output produced for the index. This parameter may be given in the form of

FORMAT={TEXT360|FORMAT|OUTPUT}

If TEXT360 is specified, the index will be generated in the form of input to TEXT360. If FORMAT is specified, the index will be generated in the form of input to FORMAT. If OUTPUT is specified (the default), the index will be generated in the form of regular output which may be copied directly to a line printer.

The WIDTH and LINES parameters may be used to specify the column width of the text and the number of lines per page of the index. These parameters may be given in the form of

WIDTH=n and LINES=n

The width defaults to 35 characters, and the lines per page defaults to 53. Included in the lines per page are blank lines inserted between alphabetic groups, such as the groups starting with "A" and "B". If TEXT360 or *FORMAT input is specified, the control lines +DEPTHnn+ and +LENGTHnn+, or WIDTH IS nn and LINES PER PAGE nn, respectively, are inserted into the index file.

For TEXT360 and FORMAT input, the COLUMNS parameter may be used to specify a multiple-column format. This parameter is given in the form of

COLUMNS=n

"n" may be 1 or 2 for TEXT360 and from 1 through 6 for FORMAT; the default is 1. Only one column is allowed for regular output.

For TEXT360, the +SWIDTHnn+ or +DWIDTHnn+ alter codes, where "nn" is the column width, and the +COLJUSTOFF+, and +NOJUST+ alter codes are inserted into the index file. For FORMAT, the WIDTH control phrase, which specifies the column width, and the NO SUPER FORMAT, CARD FIELD THROUGH 80, and NO JUST control phrases are inserted into the index file. To remove these alter codes and control phrases, the index file must be edited by hand.

The ENTRIES parameter may be used to specify the total number of entries allowed in the input to the index-generating program. This parameter is given in the form of

ENTRIES=nn

The default is 1000. This number is the total number of entries in the input, not the number of different terms in the index, e.g.,

```
2 files
5 files
```

are two entries even though they represent the same term. This parameter controls the amount of storage dynamically allocated by the program. As an indication of size, the index for MTS Volume 1, The Michigan Terminal System, has approximately 2000 entries. The example given below has 16 entries. If the number of entries exceeds the specified bound, the program will terminate abnormally with the message:

```
THERE ARE MORE ENTRIES THAN YOU SPECIFIED.
INCREASE THE ENTRIES PARAMETER IN THE "PAR=;" FIELD.
LAST CARD READ WAS:
```

Example: The example below illustrates a terminal session to construct an index. The terminal is of the 3270 type, hence the device-command control characters are entered with the "%" character. The input from the user is in lowercase; the output from the system is in uppercase.

```
#%lc
#edit -x
:insert 1
?$first page = 6
?$base = 1
?1-15 File
?1 File:sequential Sequential#files
?2 File-status
?See#also#status File-status
?3 File:changing File:changing:sequential
?  File:sequential Sequential#files
?4 MTS::#Michigan#Terminal#System
?5 "Files#and#Devices"
?11 Line#file:changing File:line File:line:changing
?  File:changing:line
?$endfile
:stop
#%uc
#empty index ok
#DONE.
#run *index scards=-x spunch=index par=;w=40 l=50 ent=40
#EXECUTION BEGINS
#EXECUTION TERMINATED
#copy index


>
> File, 6-20
>    changing, 8
>       line, 16
>       sequential, 8
>    line, 16
>       changing, 16
>    sequential, 6, 8
> File-status, See also status, 7
> "Files and Devices", 10
>
> Line file,
>    changing, 16
>
> MTS: Michigan Terminal System, 9
>
> Sequential files, 6, 8
>
#
```

Page Revised February 1979 April 1977

INDEX

April 1977

Reader’s Comment Form

FORMAT and TEXT360
Volume 15
April 1977
(January 1983 Reprint)

Errors noted in publication:

Suggestions for improvement:

Your comments will be much appreciated. The completed form may be sent to the Computing Center by Campus Mail or U.S. Mail, or dropped in the Suggestion Box at the Computing Center, NUBS, or BSAD.

Date _____

Name _____

Address _____

_____

_____

Publications
Computing Center
University of Michigan
Ann Arbor, Michigan 48109

Update Request Form

FORMAT and TEXT360
Volume 15
April 1977
(January 1983 Reprint)

Updates to this manual will be issued periodically as errors are noted or as changes are made to MTS. If you desire to have these updates mailed to you, please submit this form.

Updates are also available in the memo files at both the Computing Center and NUBS; there you may obtain any updates to this volume that may have been issued before the Computing Center receives your form. Please indicate below if you desire to have the Computing Center mail to you any previously issued updates.

Name _____

Address _____

_____

_____

Previous updates needed (if applicable):_____

The completed form may be sent to the Computing Center by Campus Mail or U.S. Mail, or dropped in the Suggestion Box at the Computing Center, NUBS, or BSAD. Campus Mail addresses should be given for local users.

Publications
Computing Center
The University of Michigan
Ann Arbor, Michigan 48109

Users associated with other MTS installations (except the University of British Columbia) should return this form to their respective installations. Addresses are given on the reverse side.

Addresses of other MTS installations:

The University of Alberta
Information Coordinator
352 General Services Bldg.
Edmonton, Alberta
Canada T6G 2H1

Information Officer, NUMAC
Computing Laboratory
The University of Newcastle upon Tyne
Newcastle upon Tyne
England NE1 7RU

Rensselaer Polytechnic Institute
Documentation Librarian
310 Voorhees Computing Center
Troy, New York 12181

Simon Fraser University
Computing Centre
User Services Information Group
Burnaby, British Columbia
Canada V5A 1S6

Wayne State University
Computing Services Center
Academic Services Documentation Librarian
5950 Cass Ave.
Detroit, Michigan 48202