CERN-DD 76-22 rev
G

CERN - Data Handling Division
DD/76/22
REVISED

Horst von Eicken
March 1978

CERN LIBRARIES, GENEVA

CM-P00059709

A U T H O R

a

Text Processing System

AUTHOR

Reference Manual

Contents

	Page
INTRODUCTION	1
THE FILE MANAGER	2
The File Directory	2
Syntax Rules	3
Command Components	4
<entry>	4
<edition>	4
<coded file>	4
File-Handling Commands	5
To Access an AUTHOR Set	5
To Create an AUTHOR Set	5
SCOPE Permanent File Commands	6
To Return and Rewind Files	6
File-Directory Commands	6
To Add a Section	6
To Delete a Section	7
To Insert a Section	7
To Change the Name	8
To Display the File Directory	8
Commands to Handle External Information	8
To Obtain Proof Copy	8
To Read Information into a Section	9
To Write Information from a Section	9
Commands to Call other Subsystems.	10
Calling the Text Editor	10
Calling the Text Processor	10
Prompt Control and Termination	10
Prompt Control	10
To End an AUTHOR System Session.	10
To Control the Field Length	10
THE TEXT EDITOR	12
Introduction	12
Lines of Text	12
Words of Text	12
The Cursor	12
The Window	12
Lines of Input	12
Delimited Strings	13
Command Syntax	13
Overall Editor Control	14
Abort Editing	14
Cursor Slots	14
End Editing	15
Jettison Last Command Line	15
Loop	15
Tab Setting for the On-line Terminal	15
The Universal Character	16
Unit of Text	16
Define Size of Verification Window	16
Cursor and Text Manipulation	16
Skip Text	17
Delete Text	18
Insert Text	18
Print Text	19
Replace Text	19

Contents

	Page
Write Text to a File	19
THE TEXT PROCESSOR	21
The Printed Page and its Formatting	21
The Title Area	22
The Text Area	22
The Footing Area	22
Commands to Shape a Page	22
To Define the Page Area	22
To Set the Margin	22
To Control Line Spacing	23
To Control Text Area Justification	23
Commands to Frame a Page	24
To Specify a Title	24
To Control the Gap	24
To Restart Page Numbering	25
To Specify a Running Footing	25
To Control the Printing of the Date	25
To Print Front Pages	25
Commands to Shape the Text	26
To Control Line Justification	26
To Define Processing of Headings	26
To Define Processing of Paragraphs	26
To Set Tabs	27
To Define a Table.	27
To Generate Table of Contents	27
Commands to Layout Text	28
To Specify and Terminate a Heading	28
To Begin a New Page	29
To Specify the Beginning of a Paragraph	29
Immediate and Hanging Indentions	29
Beginning and Ending of a Keep	29
To Insert Blank Half Lines	29
To Tab Text	30
To Set a Table	30
To Begin and End a Footnote	31
To Print Superscripts and Subscripts	31
To Underline Text	32
To Print a Marker	32
To Force Additional Blanks	32
To Print an Exclamation Mark	32
Printing Text "As Entered"	32
To Begin "As Is" Text	32
Centering Text Lines.	32
Ending Tabular Text	33
Grid Drawing Commands	33
To Begin a Grid	33
To Draw a Horizontal Line	34
To End a Grid	34
To Draw Line Segments	34
To Begin Vertical Lines	34
To Terminate Vertical Lines	35
Documentation of Programs	35
To Specify Code	35
The Reference Facility	36
To Define a Reference and its Symbol	36
To Quote References	36

Contents

	Page
APPENDIX	37
AUTHOR System Command Summary	37
File Manager Commands	37
Text Editor Commands	38
Text Processor Commands	39
Text Processing Examples	41
Page Shaping and Framing	41
Immediate and Hanging Indentions	42
Tab Setting and Grid Drawing	43
Table Processing	43
Program Documentation	46
Program Call Conventions	48
AUTHOR	48
CERNTXT	48
Availability of AUTHOR	50
Program Availability and Charging	50
Conditions of Use	50

AUTHOR

Reference Manual

INTRODUCTION

AUTHOR is an interactive text-processing system for Control Data 6000/CYBER computer systems. It has been designed to aid authors, editors, and secretaries to produce reports, letters and other documents.

The following major subsystems are contained in the AUTHOR text-processing system:

FILE MANAGER	to manage a random-access file containing the text of one or more Volumes separated into Chapters and Sections;
TEXT EDITOR	to edit a Section extracted by the FILE MANAGER;
TEXT PROCESSOR	to process information prepared by the TEXT EDITOR and extracted by the FILE MANAGER, using embedded layout and formatting commands;
POST PROCESSORS	to adapt information processed by the TEXT PROCESSOR and extracted by the FILE MANAGER to the characteristics of the different available output devices.

AUTHOR needs two permanent files for each application, one to contain the text and directory to access it, and the other one to be the log file for a session. All input from the on-line terminal is recorded on the log file, thus enabling AUTHOR to recover a session, should there be a malfunction of the software or the computer system. This log file is updated whenever information has been written to the text file. These two files, the text file and the log file, will be subsequently referred to as the "AUTHOR Set". Whenever such an AUTHOR Set is accessed, AUTHOR will check the log file to see, whether the previous session has terminated gracefully. If not, it will issue the message:

```
Text file not up to date, do you want recovery?
Please answer with:   yes / no / echo :
```

If the answer is "no", no recovery action will be taken and the file will be restored. If the answer is "yes", AUTHOR will read the log file and execute all commands that have been logged there from the interrupted session. If the answer was "echo", AUTHOR in addition to executing all commands that have been logged will echo all commands and the corresponding replies on the terminal, as if they were entered there. The recovery will be terminated by the message:

```
End of recovery.
```

The development of AUTHOR was started in 1974/75 during a sabbatical year at the Computation Center of the University of Texas at Austin and continued in the Data Handling Division at CERN. The project definition and implementation owes much to the close collaboration with the staff of both computer centres; I should like to express my thanks and appreciation to my colleagues for their help, understanding and patience. Most of all, however, I should like to thank the management of both computer centres for the opportunity given to me and their continuous interest in the project.

Note:

- At CERN, AUTHOR uses the "Rubout" key to delete the last input character. The "Backspace" key will backspace one character, but not delete it.
- The Appendix contains several layout examples to illustrate the use of Text Processor commands.

THE FILE MANAGER

File Manager commands allow the creation, maintenance, updating and inspection of the AUTHOR system text file. The commands described in this chapter can only be used when the File Manager is in control. This is always the case at the beginning of a session and whenever one of the other subsystems, such as the Text Editor or Text Processor, have finished operation.

The File Directory

The user may consider the information in his AUTHOR text file as a single string of characters. Large files, and certainly all files of more than 30000 characters, create some problems of response time and computer efficiency. Such files must be subdivided into several parts, called "Sections" in AUTHOR. Sections should be kept below some 10000 characters in order to shorten the response time of the computer.

AUTHOR holds a "File Directory" in which Sections are entered in their logical order. A listing of the file directory gives one or more lines for each Section.

The user may choose to subdivide the sequence of Sections in his file into groups of consecutive Sections. Every such group of Sections is called a "Chapter". In addition to each Section entry, the file directory holds one further entry for every Chapter. The Chapter entry immediately precedes the first Section entry in the Chapter. A listing of the file directory gives one line for each Chapter entry.

The user may choose furthermore to subdivide the sequence of Chapters in his file into groups of consecutive Chapters. Every such group is called a "Volume". In addition to the Section and Chapter entries, the file directory holds a further entry for every Volume. It precedes the first Chapter entry in the Volume. A listing of the file directory gives one line for each Volume entry.

Even when the user does not group his Sections into Volumes or Chapters, AUTHOR insists on grouping his Section, or Sections, into a single Chapter, and this Chapter into a single Volume. Thus, the very first entry of every directory is always a Volume entry, and the second entry of the directory is always a Chapter entry.

AUTHOR tags all entries in the file directory by unique numbers which are assigned [in ascending order] as soon as the entries are created. A listing of the directory shows all entry numbers in the leftmost column. Since the chronological order in which directory entries are generated by the user does not always correspond to the logical order of the text, - a Section might be inserted between two existing ones - the entry numbers, as listed in the directory, will not always be in numerical order.

Every Section, every Chapter, and every Volume has a "name". Every name is a string of 1 to 21 characters. The names of all Volumes must be unique. The names of all Chapters of any one Volume must be unique, and the names of all Sections of any one Chapter must be unique. AUTHOR maintains up to three "Editions" of a single Section. These Editions represent the text of a Section in its various stages of editing. When there are three Editions, the most recent Edition is the third, and the oldest is the first. The following rules are applied:

- The first input to a Section creates the first Edition.
- Editing of the first Edition creates a second Edition. Existing second and third Editions are automatically deleted.
- Editing of the second Edition creates a third Edition. An already existing third Edition is automatically deleted.
- Editing of a third Edition has the following effect:
  . The first Edition is deleted.

. The second Edition is re-named first Edition.
. The current third Edition is re-named second Edition
. A new third Edition is created.

An example of a file directory listing is shown in Fig. 1 below. The size column allows an estimate of the amount of text stored for a particular Section. A Section of size 5 amounts to some 2000 characters of text. In reality it is a measure of the disk space used to store the text.

Entry	Volume	Chapter	Section	Edition	Size	Date	Time	User
1	Author System Manual							
2		Introduction						
3			1					
				1. Edit	5	18/08/76	16.16	EICKEN
				2. Edit	5	19/08/76	10.19	EICKEN
				3. Edit	5	19/08/76	11.22	EICKEN
4		A First Session						
5			1					
				1. Edit	6	18/08/76	16.33	EICKEN
				2. Edit	6	19/08/76	10.53	EICKEN
				3. Edit	6	19/08/76	11.25	EICKEN
6			2					
				1. Edit	5	18/08/76	16.42	EICKEN
				2. Edit	3	19/08/76	11.11	EICKEN
7			3					
				1. Edit	2	19/08/76	11.13	EICKEN
				2. Edit	2	19/08/76	11.26	EICKEN
				3. Edit	2	19/08/76	11.28	EICKEN
8			4					
				1. Edit	1	19/08/76	11.27	EICKEN

Fig 1. Sample File Directory

Syntax Rules

. The File Manager will prompt the user with a "#" sign when it is ready to accept the next command.

. Each command is terminated with a <CR> ("carriage return"). The user can request a continuation line to the command line by ending his input line with a plus sign ("+") immediately followed by a <CR>. In this case the File Manager will prompt with a plus sign until the command has been terminated or cancelled.

. Any command can be cancelled using the back slash ("\") followed by a <CR> anywhere in the command definition.

. The following punctuation marks have special meaning in the command syntax and should not be used as separators or as part of a name given to a Volume, Chapter or Section:
the equals sign ("=")
the back slash ("\")
the period ("."

. Command verbs, file definitions and key abbreviations can be written with upper-case or lower-case letters or any mixture thereof.

. Names given to a Volume, Chapter or Section must be at least one and at most 21 characters long.

. Any punctuation mark, except the reserved ones, can be chosen as separator. Once selected, it has to be used throughout the whole command definition (i.e. separators cannot be mixed in a single command). Throughout this chapter a comma (",") has been used as separator in syntax

definitions and examples.

. In the specifications that follow, a pair of rectangular brackets ("[ ]") is used to enclose optional components. Whenever there is a choice possible for one component out of many, a pair of angular brackets ("{ }") encloses them and a slash ("/") separates them.

Command Components

Each File Manager command consists of a verb, possibly followed by one or more phrases, where each phrase consists of one or more key expressions. The following phrases are used to describe the commands:

<entry>

<entry> describes an entry in the file directory. It can either be an <entry name> or an <entry number>.

The entry name of a Section consists of

(i) the name of the Section
(ii) the name of the Chapter to which the Section belongs
(iii) the name of the Volume to which the Chapter belongs

Some commands to AUTHOR identify a Section by its entry name as follows:

```
<entry name> ::= [V=<volume name>,]
                 [C=<chapter name>,]
                  S=<section name>
```

If the Volume name is not specified, then the last Volume referenced by <entry> is implied. If the Chapter name is not specified then the last Chapter referenced by <entry> is implied. If the Volume name is specified explicitly, then a Chapter name must also be specified explicitly. Each loading of the file via the "ACCESS" command (see below) implies a reference to the very first Volume, Chapter and Section.

The entry number of an entry in the file directory identifies either a Section, a Chapter or a Volume. Some commands to AUTHOR identify a part of the text as follows:

```
<entry number> ::= #<n>
```

where n is the number that the directory gives for the Section, Chapter or Volume.

<edition>

The <edition> describes which of the various Editions of an edited Section should be addressed by the command. An Edition is specified by an expression of the type "E=<edition>" where the key codes

```
<edition> ::= / 1 / 2 / 3 / L / * /
```

have the following meaning:

1 for the first Edition
2 for the second Edition
3 for the third Edition
L for the latest Edition
* for all Editions

Whenever this parameter is optional and omitted, the latest Edition will be chosen by default.

<coded file>

The <coded file> parameter describes the name of a coded file to which information should be written or from which information should be read. The keywords used to describe the coded file are:

<coded file> ::= { A=<file name> / D=<file name> }

where A=<file name> is used if the file contains or should contain information in ASCII, and D=<file name> is used if the file contains or should contain information in Display Code.

The following file names are reserved:

"TTY"

meaning the user's on-line terminal, and

"HPLEFT" or "HPRIGHT"

indicating the left hand or right hand cartridge cassette if the user is connected via a Hewlett Packard 2644A Mini Data Station.

Whenever this parameter is optional and omitted,

A=TTY

is assumed by default.

File-Handling Commands

All file-handling commands in general follow the syntax rules of the SCOPE operating system; any exceptions will be noted with the particular command. One general exception concerns the ID=<name> parameter of the SCOPE permanent file commands. Once it has been defined in a command, it can be omitted from all other commands within the same AUTHOR system run.

To Access an AUTHOR Set

ACCESS,<permanent file name>[,<parameter list>]<CR>

To access an AUTHOR set that has been created by a previous run. The File Manager will use the <permanent file name> given with the command and construct two permanent file names by preceding this name with "AUTHORSTXT" and with "AUTHORSLOG". It will then perform ATTACH commands for both files using the ID=<name> parameter given. The local file name for the text file will be "TXTFILE", while the one for the log file will be "LOGFILE".

Note:

- The <permanent file name> and <parameter list> follow the same rules as for the ATTACH command in the SCOPE operating system. The permanent file name however is restricted to thirty characters and a local file name cannot be specified.
- The ACCESS command automatically saves and closes any other AUTHOR set that might be in use at this moment.

To Create an AUTHOR Set

CREATE,<permanent file name>[,<parameter list>]<CR>

To create an AUTHOR set consisting of two permanent files, the text file and the log file. The File Manager will generate two local files, the names of which are "TXTFILE" and "LOGFILE". It will then request from the user the names of the first Volume, Chapter and Section which he wants to generate. A file directory will be constructed with this information and written to the text file. The File Manager will then write index information to both files and catalog them under the current ID, and close them. The permanent file names used to catalog both files are constructed by preceding the supplied <permanent file name> with "AUTHORSTXT" and with "AUTHORSLOG".

Note:

- The <permanent file name> and <parameter list> follow the same rules as for the CATALOG command in the SCOPE operating system. The permanent file name however is restricted to thirty characters and a local file name cannot be specified. A multi-access permission should not be specified by the user, since AUTHOR does not contain procedures to assure correct updating for multi access operation.
- The CREATE command automatically saves and closes any other AUTHOR set that might be in use at this moment.

SCOPE Permanent File Commands

The following SCOPE permanent file commands can be used in the same way as they are described in the SCOPE Reference Manual, Chapter 5.

```
ALTER,{<lfn>/<pfn>}[,<pf keys>].
ATTACH,{<lfn>/<pfn>}[,<pf keys>].
CATALOG,{<lfn>/<pfn>}[,<pf keys>].
EXTEND,{<lfn>/<pfn>}[,<pf keys>].
GETPF,{<lfn>/<pfn>}[,<pf keys>].
PURGE,{<lfn>/<pfn>}[,<pf keys>].
RENAME,{<lfn>/<pfn>}[,<pf keys>].
SAVEPF,{<lfn>/<pfn>}[,<pf keys>].
```

It is not necessary to specify the ID parameter together with the command since AUTHOR will automatically supply the ID in use so far. It is of course possible to specify another ID in the usual way, but the user should be aware, that this new ID will then be used as default for all subsequent commands needing an ID parameter.

Note:

- These SCOPE permanent file commands should not normally be used for AUTHOR sets.
- To purge an AUTHOR set, one has to purge the text file and the log file with two separate PURGE commands.

To Return and Rewind Files

```
RETURN,<list of file names>
REWIND,<list of file names>
```

The two SCOPE commands "RETURN" and "REWIND" can be used as described in the SCOPE Reference Manual, Chapter 4.

File-Directory Commands

To Add a Section

```
ADD,<entry name><CR>
```

To add another Section to the file directory. The File Manager will check in the file directory to find out whether the Volume, Chapter or Section name specified with the ADD command is new. If so, it will add the new Section as the last entry in the appropriate part of the file directory. Specifically:

- If it was a new Section in an existing Chapter, a new Section entry will be created as last Section in this Chapter.
- If it was a new Chapter and Section in an existing Volume, a new Chapter entry will be created as last Chapter in the Volume with a new Section entry as first Section in the Chapter.
- If a new Volume name was used in the command, new Volume, Chapter and Section entries will be created. The Volume will be added as last Volume to the list of Volumes.

Example:

ADD,C=Editor Commands,S=1<CR>

would add the Chapter "Editor Commands" with Section "1" into our Volume "AUTHOR System Manual" as a new Chapter following the Chapter "A First Session" in our example in Figure 1.

To Delete a Section

DELETE,<entry>,E=<edition><CR>

To delete a Section or an Edition of a Section from the file directory, and delete the corresponding text as well. The File Manager will verify the existence of the Section in the file directory and delete it according to the Edition specification given. The following rules are applied:

- If the entire Section entry should be deleted, the key expression "E=*" ("all Editions") should be used.
- If the first Edition is to be deleted, an existing second Edition will be renamed first Edition, and an existing third Edition will then be the second Edition. If no Edition except the first one exists, the entire Section entry is deleted.
- If the second Edition is to be deleted, an existing third Edition will be renamed second Edition.
- If the latest Edition is to be deleted, and this Edition is the first Edition of a Section, the command will be ignored.
- If the <entry> specification denotes an entire Chapter, all Sections of that Chapter will be deleted accordingly.
- If the entry specification denotes an entire Volume, all Chapters and Sections of that Volume will be deleted accordingly.

Following the execution of the command the File Manager will request the approval of the user before accepting the modifications as final. If the user denies it, the old directory will be restored. This should give an additional safeguard against accidental deletion of Sections.

Example:

DELETE,#4,E=1<CR>

would delete the entire Chapter "A First Session", while

DELETE,#7,E=3<CR>

would delete the third Edition of the third Section in Chapter "A First Session" of our example in Figure 1.

To Insert a Section

BEFORE,<entry number>,<entry name><CR>
AFTER,<entry number>,<entry name><CR>

To insert the Section <entry name> before or after the entry specified by <entry number>. The File Manager will verify the existence of the entry <entry number>, check that the Section <entry name> does not yet exist and insert it with the given name into the file directory before or after the entry <entry number> and allocate a new entry number for the inserted Section. It will furthermore check that no duplication of Volume names within the AUTHOR set, no duplication of Chapter names within a Volume and no duplication of Section names within a Chapter can occur.

If a new Chapter name was used in <entry name> and the <entry number> refers to a Chapter, a new Chapter entry will be created. If a new Volume name was used in <entry name> and the <entry number> refers to a Volume, a new Volume entry will be created.

Note:

. The command cannot be used to split Volumes or Chapters, i.e. one can neither insert a new Volume between Chapters of an existing Volume, nor a new Chapter between Sections of an existing Chapter.

Example:

BEFORE,#2,C=Abstract,S=1<CR>

would insert the new Chapter "Abstract" before the Chapter "Introduction" into the Volume "AUTHOR System Manual" of our example in Figure 1.

To Change the Name

NAME,<entry number>,<entry name><CR>

To change the name of a Volume, Chapter or Section entry specified by <entry number>. The File Manager will verify the existence of the entry <entry number>, check that the new name does not yet exist and change the old name.

Example:

NAME,#1,V=Author System<CR>

would change the Volume name of our example in Figure 1 from "Author System Manual" to "Author System".

To Display the File Directory

DIRECTORY[,<entry>][,E=<edition>]<CR>

To display all or part of the file directory.

Example:

DIRECTORY<CR>

will display the file directory for the entire AUTHOR set, while

DIRECTORY,#4,E=*<CR>

would display the file directory for the Chapter "A First Session" of our example in Figure 1.

DIRECTORY,E=L<CR>

should be used if one just wants to see a file directory where only the latest Edition of each Section is displayed.

Commands to Handle External Information

To Obtain Proof Copy

PROOF[,<entry>][,E=<edition>][,<coded file>]<CR>

To prepare a proof copy from one or more Sections in the AUTHOR SET. The PROOF command will write the selected Section, Chapter or Volume to the specified file in the code requested. Each Section of text will start a new portion of output, which will be preceded by a header specifying the AUTHOR SET, user name, date and time as well as entry number, Volume, Chapter and Section name.

Single Sections are specified by either their entry name or their entry number, while an entire Chapter or Volume can only be specified by its entry number.

Example:

PROOF,C=File Manager Commands,S=5<CR>

will write the contents of the latest Edition of the fifth Section in Chapter "File Manager Commands" from the current Volume to the on-line

terminal. The default A=TTY will be used, since the optional <coded file> parameter has been omitted.

To Read Information into a Section

READ,<entry>[,E=<edition>][,<coded file>]<CR>

To read information from a coded file into an empty Section. The File Manager will verify the existence of the entry and accept the request if the Section is empty. Reading will commence at the current position of the file and terminate with the end of the current record. The information read will be converted to ASCII code if necessary and stored in the specified Section.

It is possible to read information into an entire Chapter or Volume by specifying the entry number of the Chapter or Volume in the READ command. Information will be read into all Sections, provided they are empty. Reading will be stopped when an attempt is made to read into a non-empty Section, a message will be given and the rest of the command ignored. A new Section will be started whenever an end of record is encountered on the input medium. The occurrence of an end of file or an end of information will also terminate the reading process and the rest of the command will be ignored.

Example:

```
READ,V=Author System Manual,+<CR>
C=Text Editor Commands,S=1,A=HPLEFT<CR>
```

The above READ command has been entered as a multi-line command, using the plus sign ("+") as last symbol prior to the <CR> to request a continuation line. Reading is requested into the first Section of the Chapter "Text Editor Commands" in the Volume "Author System Manual". The file name used for the coded file, "HPLEFT", is one of the reserved file names and indicates to the File Manager that the user is connected via a Hewlett Packard Mini Data Station and that reading should be from the left-hand cartridge of the terminal in ASCII code.

To Write Information from a Section

WRITE[,<entry>][,E=<edition>][,<coded file>]<CR>

To write information from a Section to a coded file. The File Manager will commence writing at the current position of the coded file. The information of the Section is converted if necessary and written as one record to the file. If the Section is empty, an end of record marker will be written. It is possible to write an entire Chapter or Volume to the file by specifying the entry number of the Chapter or Volume in the WRITE command. Each Section will be written as one record.

The file will remain positioned after the last record written.

Examples:

```
WRITE<CR>
```

will write the latest Edition of each entry in the AUTHOR SET to the on-line terminal.

```
WRITE,V=Author System,C=Editor Routines,+<CR>
S=Cursor Move,E=L,D=COMPILE<CR>
```

will write the latest edition of the Section "Cursor Move" in Chapter "Editor Routines" from the Volume "Author System" to the file "COMPILE" converting it to Display Code.

```
WRITE,#4,E=L,A=TEXT<CR>
```

will write the latest Edition of each Section in Chapter "A First Session" of our example in Figure 1. to the file "TEXT" in ASCII code.

Commands to Call other Subsystems.

The File Manager is always in control whenever a session starts. All other subsystems such as the Text Editor and Text Processor are called in from the File Manager and will return to the File Manager.

Calling the Text Editor

EDIT,[,<entry>][,E=<edition>]<CR>

When calling the Text Editor, a Section is selected for editing. The File Manager will read the contents of the Section into core and will transfer control to the Text Editor. If <entry> specifies a Chapter or Volume rather than a particular Section, the very first Section of that Chapter or Volume will be selected for editing.

Example:

EDIT<CR>

will select the latest Edition of the first Section in the first Chapter of the first Volume for editing.

EDIT,S=Layout Commands,E=1<CR>

will select the first Edition of the Section named "Layout Commands" in the current Chapter and Volume for editing.

Calling the Text Processor

(Syntax is not yet decided)

The version of AUTHOR described here does not yet contain its own text processor. There is however a stand-alone version available. The Chapter "The Text Processor" describes it in detail. The command structure described for the stand-alone processor will be upward compatible with the one being developed for the text processor in the AUTHOR system. Using the "WRITE" command of the File Manager the user can create a local ASCII file of his choice and use it as input to the stand-alone text processor.

Prompt Control and Termination

Prompt Control

BRIEF<CR>
VERBOSE<CR>

The amount of prompting done by the AUTHOR system can be controlled by the above two commands. By default the AUTHOR system will be verbose and initial users are strongly advised to stay in this mode for their first sessions. When "BRIEF" is selected, the messages are restricted to their bare minimum.

To End an AUTHOR System Session.

END<CR>

To end the current AUTHOR system session. The File Manager will save and close the AUTHOR set that might be in use and return control to the operating system.

To Control the Field Length

#n #

AUTHOR allows the user to check and control the maximum amount of field length allocated to his session. The "#" command can be used under the File Manager as well as under the Text Editor. Typing "#" the user will receive the following reply:

```
Current size settings are:
ACTUAL SIZE    USER SIZE    MAXIMUM SIZE
  021476         040000         060000
```

indicating that AUTHOR currently occupies 21476 octal locations of core to hold the program and text. The user allows AUTHOR to extend the field length, if necessary, up to 40000 octal locations. The maximum size of 60000 octal locations is allocated at installation time of AUTHOR and can only be modified by changing the parameter in the storage manager of AUTHOR.

The user now can modify the user size by typing "#n", where n must be an octal number between actual size and maximum size. AUTHOR will round the new user size to be a multiple of 100 octal.

Example:

```
#23457
```

will cause the following message:

```
New size settings are:
ACTUAL SIZE    USER SIZE    MAXIMUM SIZE
  021476         023500         060000
```

allowing AUTHOR to extend the field length up to 23500 octal locations.

THE TEXT EDITOR

Introduction

Text Editor Commands allow the insertion, modification, and deletion of text in a Section.

Lines of Text

The Text Editor knows its Section of text as a sequence of "lines". A line boundary occurs after every <CR>; i.e. each line includes exactly one <CR> and this is the last character of the line. AUTHOR ensures that the last character of a Section is always a <CR>. Lines may consist of at most 180 characters.

Words of Text

The Text Editor knows each line as a sequence of words. A word boundary occurs within a line wherever a blank is followed by a non-blank character. The blank is the last character of the preceding word and the non-blank character is the first character of the word which follows. AUTHOR ensures that each <CR> is preceded by a blank; thus the <CR> always constitutes the last word of the line.

The Cursor

The Text Editor maintains a pointer, called the "cursor", which identifies a character boundary somewhere in the text. When a new Section of the Volume is passed to the editor, the cursor is positioned in front of the first character of the Section. The cursor is then repositioned by the Text Editor in response to commands which are given by the user. Moreover, what the Text Editor will do in response to a command from the user depends, for most commands, on the current position of the cursor. A clear understanding of how the Text Editor manipulates the cursor is therefore essential to an understanding of the Text Editor commands.

The Window

Whenever the Text Editor has executed a command line it tells the user where the cursor is now positioned: it outputs a "window" of text - i.e. a character string which surrounds the new position of the cursor, with the character asterisk ("*") inserted at the point of the cursor. The user may control the size of the window with the V-command given below.

Lines of Input

The Text Editor accepts input from the terminal one line at a time. The user terminates his line when he presses the <CR> key. When the Text Editor has processed the line of input it sends out a "prompt character" to the terminal, thus requesting a further line of input from the user.

There are two kinds of input line:

(i) data lines and
(ii) command lines.

Whenever the editor expects a data line from the user it prompts with the character plus ("+"). When the editor expects a command line from the user it prompts with the character colon (":").

Data Lines

The Text Editor inserts an "input string" into the text for every data line. The input string is entered at the current position of the cursor; the new position of the cursor will be immediately after the last character of the input string.

The input string that the Text Editor inserts in the text depends on the character that precedes the <CR> on the data line.

1. If the character in front of the <CR> is a blank, then the input string that is stored is identical with the data line; it includes the <CR>.
2. If the character in front of the <CR> is not a blank and not the ASCII control character "end of text"[1], <EOT>, then the input string that is stored consists of all characters of the data line up to, but not including, the <CR>, followed by a blank, followed by <CR>.
3. If the character in front of the <CR> is <EOT> then the input string that is stored consists of all the characters of the data line (if any) up to, but not including, the <EOT>.
4. If the data line consists of the text string "%EOR<CR>"[2] then the input string that is stored consists of a <CR> or a blank followed by a <CR>, obeying rules 1. and 2. above.

Following the input of a data line, the Text Editor will prompt the user for a command line if the data line just processed was terminated by <EOT><CR> or consisted of the text string "%EOR<CR>"; otherwise the user will be prompted for another data line.

Command Lines

A command line consists of one or more Editor commands. Each command consists of a sequence of non-blank characters (delimited strings are the only exception). If there is more than one command in a command line then successive commands must be separated by one or more blanks. Blanks may precede the first command and blanks may follow the last command of any command line.

If more than one command appears in a command line then the Text Editor processes them from left to right.

Delimited Strings

Two types of character strings may form part of an editor command: search strings and replacement strings. No such string can include the character <CR> or exceed 50 characters in length. The null string is acceptable as a replacement string, but not as a search string.

String Delimiters

Each string must be preceded and followed by the same string delimiter. A delimiter can be any punctuation mark, provided it is not + or - or (or the universal character (see below, in chapter "Overall Editor Control"), and provided it does not appear in the string itself.

Command Syntax

Each Text Editor command consists of a "command identifier", possibly followed by one or more values of "parameters" of the command. Any parameter values follow the command identifier and each another without

---

[1] The "end of text" control character <EOT> is generated by pressing simultaneously the "Control" key and the key for the letter "d" or "D" on the keyboard.

[2] The text string "%EOR<CR>" is interpreted as an "end of record" request by SUPERMUX at CERN and will cause AUTHOR to read an "end of record" from the terminal.

intervening blanks or separators. The parameter values of a command must be specified in a fixed order. However, the specification of values is optional for some parameters; in such cases, if a value is omitted then the Text Editor will supply a default value for the parameter. Text Editor commands are available to define the default values for optional parameters.

As an example, consider the command

```
D3W     (delete 3 words of text)
```

Here D is the command identifier of the delete command, W (words, not characters or lines) is the value for the parameter "unit of text" and 3 the count of units to be deleted. If the count 3 were omitted (i.e. DW) then the Text Editor would supply a count of 1 as default value. If the unit specification W were omitted (i.e. D3) then the Text Editor would assume whatever default value is currently defined for the parameter "unit of text". The default value will be C (character) as long as the user does not redefine the default; the command D3 will then delete 3 characters. But if the user has redefined the Editor´s default value for the parameter "unit of text" with the command

```
UL      (retain "lines" as default unit of text),
```

then the command D3 will now delete 3 lines.

All command identifiers are single latin letters, and so are some values of parameters. We shall denote these values always with capital letters and shall always use small letters to denote the names of parameters. Thus u will denote the parameter name "unit of text", and C, W, and L will denote the three values which may be given to the parameter u. However, the Text Editor accepts capital and small letters as command identifiers and parameter values without distinction and leaves the choice to the user. DW, dw, Dw, and dW do the same job.

Overall Editor Control

Overall control over the Text Editor can be used to modify the various defaults used, to control its command execution and prompting, and to terminate editing of a Section.

Abort Editing

A

The Text Editor returns control to the File Manager. The File Manager retains the old Editions of the present Section; no new Edition is stored.

Cursor Slots

K

The Text Editor remembers twelve positions, called "cursor slots", B, E, S0, S1, ..., S9. B and E always point, respectively, to the beginning and to the end of the Section. The remaining slots may point to any position of the Section.

The ten slots S0 to S9 are set to the beginning of the Section whenever the File Manager passes a Section of text to the Editor. The position of these slots may be redefined as follows:

K Kd

where d= 0, 1, ..., 9

K is equivalent to K0.

The current position of the cursor will be remembered as cursor slot Sd until the position of slot Sd is redefined with the K-command, or until another Section of text is passed to the Text Editor.

End Editing

E E+ E-

The Text Editor returns the present Section of text to the File Manager. A new Edition will be stored. (See "File Directory" in the chapter "The File Manager" for details.)

E returns control to the File Manager.

E+ passes the next Section of the Chapter to the Text Editor. If there is no subsequent Section in the Chapter, the File Manager will create a new Section. The File Manager will generate a new name if the existing Sections of the Chapter are named with natural numbers (1, 2, 3, ...), or else the user will be asked to specify a name for the new Section.

E- passes the preceding Section of the Chapter to the Text Editor. If the Section returned was the first Section of a Chapter then control returns to the File Manager.

Jettison Last Command Line

J

The command must be the only command of the command line.

Text and cursor are restored to what they were prior to the execution of the preceding command line. The current status of the files is not altered by the J command. (A second J command in sequence has no effect at all.)

Loop

L Ln

where n = 1, 2, ...

The command may appear only once in a command line and may not be the last command of the line.

Using Ln, the command line is executed as if the sequence of commands that follows the command Ln had been specified n times. (L1 has therefore no effect on the execution of a command line.)

Using L, the command line is executed as if the sequence of commands that follows the command L had been repeated an infinite number of times.

Infinite loops should be used with caution since AUTHOR may never come out of them. Consider the example:

UC I/xyz/ S-2 L S/z/ S-1/x/

Tab Setting for the On-line Terminal

T $T,n_1,n_2,...$

T

Up to nine tab positions can be defined for the on-line terminal. The range of their positions depends largely on the type of terminal used. They are defined to the Text Editor as follows:

$T,n_1,n_2,...$ where $n_1,n_2,..$ are up to nine unsigned positive integers in ascending order, each one preceded by a comma.

T is equivalent to T,11,20,36

$T,n_1,n_2,...,n_9$ sets tabs at positions n_1, n_2, and so on; if less than 9 positions are specified with the command then the remaining tabs are set to position 1. (T,1 therefore clears all tab settings.)

The default is a tab at position 7.

Note: The tab handling for the on-line terminal is installation dependant. At CERN the Text Editor will inform the operating system software (SUPERMUX) about the desired tab setting. Pressing the tab control character on the keyboard will then cause spacing of the carrier and insertion of blanks into the line of input.

The Universal Character

C Cs

where s may be any special character.

A "universal character" may be included in search strings, to match any character except the <CR> in text comparisons.

The command C declares the universal character as undefined, while Cs defines s as the universal character. The definition will be valid until a further C command is executed.

The star ("*") is used as the default universal character.

Unit of Text

Ut

where t = C, W, L

Positioning of the cursor may depend on the "unit of text" parameter which is used during the execution of the command. The units of text known to the Text Editor are lines (L), words (W), and characters (C). Whenever the parameter is omitted in commands, a default value will be used by the Text Editor. Its initial value when AUTHOR is first called is set to character (or C) as unit of text.

The command Ut defines t as the default value of the parameter "unit of text". The default value will be used by the Text Editor whenever a value for unit of text is omitted in subsequent commands.

Define Size of Verification Window

V0 Vnt

where n= 1, 2, 3, ...

t= C, W, L, or omitted

The Text Editor can display a "window" of text around the new position of the cursor whenever a complete command line has been executed. In addition, within a command line it will display the window whenever the cursor is moved to another line and the previous line has been modified.

The size of the window is the entire line when the Text Editor is first called. This is also the maximum size of the window. The user may suppress the display of windows, or modify the window size, as follows:

V0 suppresses the display of the window.

Vnt enables the display of a window. The window will start n units of text to the left of the cursor position or at the start of the current line, whichever position is nearer to the cursor. The window will end n units of text to the right of the cursor position or at the end of the current line, whichever position is nearer to the cursor.

Cursor and Text Manipulation

Cursor and text manipulation commands allow the user to insert text, delete text, replace text and to selectively write text to local disk files.

Skip Text

SB SE SSd Sit St/old/ Sit/old/

where

B, E, Sd	are any cursor slots,
/old/	is a delimited string,
t	is a unit specification, and
i	is either omitted, or is the plus ("+") sign, or the minus ("-") sign, or any signed or unsigned integer; i must not have the value zero in the form Sit/old/.
	If i is + or is omitted then 1 will be assumed. If i is - then -1 will be assumed.

SB, SE, SSd bring the cursor to the position of the specified cursor slot; e.g. SE brings the cursor to the end of the Section.

S0C leaves the cursor unchanged. So do S0W (and S0L) provided the cursor is currently at a word (or line) boundary. If not, then S0W (and S0L) move the cursor to the beginning of the unit of text inside which it is currently positioned.

Sit (i= 1, 2, 3, ...) positions the cursor i units of text to the right of the position that is attained by S0t, provided the Section of text extends that far to the right; otherwise Sit is equivalent to SE.

S-it positions the cursor i units of text to the left of the position that is attained by S0t, provided the section of text extends that far to the left; otherwise S-it is equivalent to SB.

Note that the command S7W is equivalent to the command line

S0W L7 SW

The command St/old/ moves the cursor successively forwards to the positions that would be attained with S0t, St, S2t, S3t, ... The process stops as soon as the text to the left of any such position is the string /old/. (By definition, any universal character in string old matches the corresponding character in the text.) If a match is found after n failures and string /old/ consists of k characters then St/old/ is equivalent to the command line

Snt SkC

i.e. the cursor is positioned to the <u>right</u> of the matching string.

The command S-t/old/ moves the cursor backwards to the first of the positions S-nt (n= 0,1,...) at which the text to the right matches old. Given n failures, S-t/old/ will be equivalent to S-nt; i.e. the cursor is positioned to the <u>left</u> of the matching string.

The command Sit/old/ is equivalent to the execution of the command line

Li St/old/	if i > 0

and

Lm S-t/old/	where(m=-i) if i < 0, i.e. it looks for the i-th match.

Examples:

Consider the effect of each of the following skip commands on the text:

Words are our precision* tools. <CR>

(with the cursor position as shown by the asterisk).

S-2W:	Words *are our precision tools. <CR>
S-W:	Words are *our precision tools. <CR>

```
S0W:      Words are our *precision tools. <CR>
SW:       Words are our precision *tools. <CR>
S2W:      Words are our precision tools. *<CR>
S3w:      Words are our precision tools. <CR>*
S-2C/e/:  Words ar*e our precision tools. <CR>
SC/o/:    Words are our precision to*ols. <CR>
```

Delete Text

Dp D(f)p

where

p — is any of the parameter specifications that are defined for the Skip Text command to position the cursor

f — is any SCOPE file name, optionally preceded by a minus ("-") sign. (A minus sign specifies REWIND before data transfer.)

The delete command moves the cursor from its current position to the same position that would have been attained with the parameter p in a skip command. All characters between the two positions are then deleted from the text. They are copied as one logical record to a file, if a file name has been specified with the delete command.

An extra blank is appended to the first part of the remaining text, (if any,) if the second part begins with <CR> and the first ends with a non-blank character. If the second part is empty and the first does not end with a <CR> then the second part is assumed to be the character <CR>.

Any cursor slot which pointed to a position inside the text that has been deleted, or to either end of that text, is left pointing between the two parts of the remaining text.If the entire Section has been deleted, all cursor slot positions point to the end of the Section.

Insert Text

I/new/ I(f) I

where

/new/ is a delimited string of from 0 to 50 characters,

f — is any SCOPE file name, optionally preceded by a minus ("-") sign. (A minus sign specifies REWIND before data transfer.)

I/new/ — The string is inserted at the current position of the cursor. The command is aborted if the line extended by the command would become longer than 180 characters. An extra blank is appended to the inserted string in the case where the cursor is positioned in front of a <CR> and the string terminates with a non-blank character.

I(f) — will optionally rewind the file before the next logical record is transferred into the Section. All text is inserted at the current cursor position. Lines extending to more than 180 characters are truncated and a message given. The end of the logical record is interpreted as a data line containing only %EOR<CR>. (see "Data Lines" above).

I — will prompt the user for a data line from the on-line terminal. The input of data lines will continue until the user terminates a data line with the characters <EOT><CR> or the data line consists of the text string "%EOR<CR>" (see "Data Lines" above).

The new position of the cursor marks the immediate end of the insertion. Any cursor slot pointing to the old position of the cursor will now point to the new position of the cursor.

Print Text

PB PE PSd Pit Pit/old/

where the parameter specifications are defined as for the skip command.

The print command moves the cursor from its current position to the same position that would have been attained with these parameters in a skip command. All characters between the two positions are then printed at the terminal. The cursor is then left at its old position.

Replace Text

Rit/old/new/

where the parameter specifications of the replace command are defined as for the skip and insert commands.

The replace command is best described as a sequence of skip, delete and insert commands executed in a loop:

If the repetition count i is positive, then the action of the replace command is as follows:

Li St/old/ D-C/old/ I/new/ <CR>

If, however, the repetion count i is negative and m = -i then the action of the replace command is:

Lm S-t/old/ DC/old/ I/new/ S-C/new/ <CR>

i.e. the command will search forwards (or backwards) beginning with the current cursor position to find the next occurrence of the string /old/ at the unit of text t. If found, it will delete the string /old/ and replace it by the string /new/. If the search is forward and the repetition count not yet exhausted, the process will be repeated.

If the search is backwards, the cursor is moved in front of the inserted string before the repetition count is tested, to see whether the process should continue.

The replace command is terminated when the beginning or end of the Section is reached, even if the repetion count is not yet exhausted.

All cursor slot positions are updated as if the replace command were executed as a sequence of skip, delete and insert commands.

Note:

. If the absolute value of i is zero ("0"), all strings /old/ will be replaced by /new/. The search direction is depending upon the sign of i, i.e. "r-0/old/new/" will search backwards, while "r0/old/new/" or "r+0/old/new/" will search forwards from the current cursor position.

Write Text to a File

W(f)p

where

p is any of the parameter specifications that are defined for the skip text command to position the cursor

f is any SCOPE file name, optionally preceded by a minus ("-") sign. (A minus sign specifies rewind before data transfer.)

The write command moves the cursor from its current position to the same position that would have been attained with the parameter p in a skip command. All characters between the two positions are then written as one logical record to the file specified. The cursor is then left at its old position. The file can optionally be rewound before data transfer.

THE TEXT PROCESSOR

The Text Processor described in this chapter is a modified version of CERNTEXT[1] which was developed in 1970-71 as a pilot scheme. It has been used since then as the output processor for the CERN Library Mechanization Project. CERNTEXT is oriented towards typewriters with fixed spacing of characters and was designed following the concepts of IBM's text processing program Text 360[2]. Many of the CERNTEXT commands are copies of TEXT 360 commands in feature and description. CERNTEXT is the only Text Processor now available with AUTHOR and works as a stand-alone program.

All Text Processor commands are embedded in the text which is to be processed. The exclamation mark ("!") has been chosen to indicate to the Text Processor the beginning of command mode. The command "!!" (double exclamation mark) instructs the Text Processor to print an exclamation mark. The commands can generally be grouped into three classes:

1. General Layout Commands.

   General layout commands consist of a verb, possibly followed by one or more parameters. The command verb must be written with capital letters. A blank terminates the command string. It is not possible to combine several verbs to one command string.

2. Line-Layout Commands.

   Line-layout commands are single-letter commands, possibly followed by one or more numbers as parameters. They must be written with lower-case letters. Several line-layout commands can be combined in one command string. Each command string begins with an exclamation mark ("!") and terminates with a blank. "!pi3 " is an example of such a command string, requesting the start of a new paragraph with an indention of three characters for the first line of the paragraph.

3. Word-Layout Commands

   Word-layout commands are strictly one-character commands preceded by an exclamation mark. They are inserted into the text to obtain layout functions such as underlining, subscripting and superscripting.

The various Text Processor commands are subsequently described, irrespective of their class, in groups of logically connected commands. The Appendix contains more detailed layout examples and should be consulted, if there is no example quoted with the command description.

The Printed Page and its Formatting

The basic unit of the text processing program CERNTEXT is the printed page. Its area is defined by the number of half lines per printed page and the number of characters per line. In general each page is considered as consisting of three distinct areas:

---

1) "CERNTEXT a Mechanized Pilot Scheme for the Publishing of Manuals", CERN, Data Handling Division, DD/71/4 by H. von Eicken. (Out of print)

2) TEXT 360, W.J. LeSuer, IBM 360D-29.4.001, Second Edition (March 1969), IBM Corporation, Programming Publications, Boulder CO

The Title Area

The title area contains the text to be printed at the top of each page. It starts with the first half line of the page and might comprise several text lines. The text for the title might be interspersed with other layout commands to achieve the desired format.

The Text Area

By default the text area is separated from the title area by four blank half lines. The "GAP" command allows the user to modify this default. The text area is formatted independently from the title and foot area. Its depth in terms of half lines is governed by the overall depth of the page, the number of half lines needed for the title and footing, and for the empty half lines needed to separate the title and footing from the text area. Even if no title has been specified, the number of separation lines is deducted from the depth of the page area to obtain the depth of the text area. The text in the text area can be formatted with a left and right hand margin, thus reducing the overall number of characters per line inside the text area. A wide range of layout commands allowing definition of headings, paragraphs, footnotes, table formats etc., can be specified, to achieve various layouts.

The Footing Area

The footing area is separated from the text area by four blank half lines and contains text to be printed at the bottom of each page as running footing. It currently contains the date of processing, the page number, and, optionally, a user-defined text string as a single line of text. Unless instructed otherwise, the layout of this line will alternate between right-hand and left-hand pages.

Commands to Shape a Page

To Define the Page Area

!WIDTHw
!DEPTHd Default: w = 70; d = 120.

To set the width and depth of a page. The width of a page is measured as the number of characters per line, while the depth of a page is measured in half lines per page (twice the number of lines). The Text Processor works with half linefeeds as the basic measure to move from line to line. This enables it to cope with superscripts and subscripts on typewriters, allowing all characters to be used in superscript and subscript position.

The Text Processor will terminate the processing of the current page, if any, and print it. A new page is started with the new width and depth.

Note:

- The maximum width of a page is 120 characters per line, and the maximum depth is 240 half lines per page (120 lines).
- The defaults are set for a normal DIN A4 page which has 70 characters per line and 120 half lines per page (60 lines).
- The minimum number of characters per line is 8 while the the depth must be at least 50 half lines.

To Set the Margin

!MARGINm
!LENGTHl Default: m = 0; l = 70.

The Text Processor will always generate a left margin of four characters in addition to the specified WIDTH of a page. It uses this margin to be able to print a user requested marker in this margin (see below, "Set Marker" command). The "MARGIN" command now reduces the usable width of

the line by leaving m additional spaces blank at the start of each line. The "LENGTH" command reduces the usable width of the line still further by restricting it to l characters and leaving a right hand margin. The margin and the length of a line are both measured from the leftmost position of the line. In other words:

The "MARGIN" command leaves m spaces blank at the start of each subsequent line, while the "LENGTH" command leaves "width - l" spaces blank at the end of each line, "width" being the size of the line specified with the "WIDTH" command.

Both commands, "MARGIN" and "LENGTH" terminate the processing of the current line, if any, and begin a new line according to the new specifications.

Note:

- The default setting for the "margin" is zero while that for the "length" is equal to the "width" of the line.
- The remaining usable width to print a line must be at least 8 characters.
- A change of the width of a page will not automatically reset the length of a line, unless the new width is smaller than the current length, i.e. if the length is currently 65 and width changes from 70 to 80, length will not be modified, while a change of width from 70 to 60 would reset length automatically to 60.

To Control Line Spacing

!SPACINGs
!SINGLE
!DOUBLE Default: s = 2.

The automatic spacing of text lines can be set to a specific number of half lines, with s = 1, 2,..., 10 using the "SPACINGs" command.

Note:

- The command does not terminate the processing of the current line, if any.
- The default is set to two half lines.
- The commands "SINGLE" AND "DOUBLE" will set the line spacing to two or four half lines respectively.

To Control Text Area Justification

!COLJUST
!ECOLJUST Default: !ECOLJUST

The Text Processor terminates the processing of a page under the following conditions:

- the page depth has been reached,
- a command requesting the beginning of a new page has been encountered,
- a command implying the start of a new page has been given (e.g. !DEPTH, !TITLE, etc),

In the first of these conditions only, the Text Processor can be asked to perform an additional layout function: namely to adjust the text area in the "vertical" direction. This adjustment is called "justification" of the text area.

The Text Processor will always examine the text for occurrences of headings and the beginnings of paragraphs. It will ensure that neither headings nor paragraphs start within the last three lines of text on a page. It will also ensure that neither footnotes and their references,

nor text declared to be kept together on a page, is split between pages. This might result in fewer lines of text on a given page of text than demanded by the DEPTH command.

To avoid unbalanced pages, the Text Processor can be instructed, by the COLJUST command, to perform text area justification. It will then compute the additional number of half linefeeds needed to adjust the text according to the following rules:

- Each level-one heading may be preceded by up to two additional half linefeeds.
- If this still does not adjust the text area, the Text Processor may insert up to one half linefeed in front of each level-two heading.
- If this is still insufficient, the same may be done for each level-three heading and if necessary also for each start of a new paragraph.
- Should these operations not adjust the text area, the text will be written without being adjusted.

Unless explicitely instructed to perform justification, the Text Processor will not try to adjust the text area. The justification feature can be selectively switched on and off using respectively the COLJUST and ECOLJUST commands in the text stream.

Commands to Frame a Page

To Specify a Title

!TITLE<blank><text><blank>!ETITLE

To obtain a title on each subsequently formatted page. The Text Processor will read the text following the "TITLE" command and process it according to the layout specifications currently in use. It is possible to specify within the text of the title additional standard layout commands to format it. Title processing is terminated with the "ETITLE" command.

The "TITLE" command terminates the processing of the current page, if any, and begins a new page. This page and all subsequent ones will be started with the new title.

Note:

- The title, once formatted, will not change on subsequent pages, even if the size specifications for the page are modified.
- The title will be separated from the text by an additional four blank half lines.

Example:

```
!TITLE !c AUTHOR !s3c Reference Manual !ETITLE
```

will produce the same title as printed for this reference manual.

To Control the Gap

!GAPg Default: g = 4.

The Text Processor by default will generate four blank half lines to separate the title area from the text area, even if no title has been defined. This default can be modified with the "GAP" command and set to g half lines, with g = 0, 1,..., 10.

Note:

- The "GAP" command does not cause a new page to be started, it merely changes the default to be used when the next page is to be printed.
- The user is advised to use the "n" command, should he want to effectively control the formatting of pages with the "GAP" command.

. The "GAP" command does not simultaneously change the gap between the text area and the footing area.

To Restart Page Numbering

!ARAB(xxxxx)
!ROMAN

To restart the page numbering with Arabic or Roman numerals. The Text Processor prints the date, a page number and, optionally, a running footing at the bottom of each page. The positions of the date and page number alternate to allow recto-verso printing of the pages. The page numbering can be done either with Arabic numerals or with Roman numerals. The Roman numerals are, however, restricted to the first ten. The command does not terminate the processing of the current page, if any, but merely resets the page number to ("1") or ("i") as the case may be.

The ARAB command can optionally be followed by up to five characters enclosed in parentheses. These characters, if specified, become part of the page number printed on each page. They will precede the numeral with one interleaving blank inserted by CERNTEXT. This feature is particularly useful if each Chapter of a Volume should have its own page numbering.

To Specify a Running Footing

!RFOOT <text> !

To specify a text, not containing any layout commands, to be printed at the bottom of each page. The text follows the page number on left hand pages and precedes it on right hand pages. The text is terminated by a single exclamation mark, and its length is restricted to the current usable line width less the space to accommodate the date and page number.

Note:

. The command does not terminate the processing of the current page, if any, but merely changes the text for the running footing.

Example:

!RFOOT The Text Processor !

will produce the same running footing as printed on this page.

To Control the Printing of the Date

!DATE
!EDATE Default: !DATE

The Text Processor will print the date of processing as part of the last line of each page. This line in addition contains the page number and, if specified, the running footing. This feature can be switched off using "EDATE" and switched on using "DATE". The default is to print the date.

To Print Front Pages

!FRONT
!EFRONT Default: !EFRONT

The FRONT command should be used if only right hand pages are to be formatted by the Text Processor. In other words:

The FRONT command suppresses alternation of the position of the page number and forces it to be printed in the right hand bottom corner of each page.

The EFRONT command returns the Text Processor to print the page number in alternating positions. This is also the default.

Commands to Shape the Text

To Control Line Justification

!JUST
!EJUST Default: !JUST

When reading the stream of input text, the Text Processor will delete extraneous blanks between words and ignore all control characters such as linefeeds, carriage return, etc. It will arrange the text according to the desired line and page sizes following only the instructions of the embedded layout commands. This is the principal mode of operation for the Text Processor, the other being that in which pre-formatted text is accepted (see "a" and "c" commands below).

The Text Processor will justify lines of text within the boundaries specified by the various commands controlling the size of the printable line of text. It will add additional blank spaces between the words of a line to make lines flush with the left- and right-hand margins. This line justification feature can be switched on using "JUST" and switched off using "EJUST". The default is justification of lines.

To Define Processing of Headings

!HEADING(h_1=s_1,h_2=s_2,h_3=s_3)

To ensure a consistent layout throughout a document, the Text Processor can be instructed by the HEADING command to automatically precede each level of heading in the text (not in the table of contents) with a certain number of half linefeeds. The default number is the spacing value chosen. (Single or double spacing of lines). The HEADING command now allows the user to selectively change this default for the three levels of heading. The parameter h_i in the HEADING command specifies the level of the heading while s_i defines the number of half linefeeds to be issued, whenever such a heading is encountered.

```
!HEADING(1=6,2=4,3=3)
```

for instance specifies automatic spacing for all three levels of heading, namely level-one headings to be preceded by 6 half linefeeds (or three linefeeds), level-two headings by 4 half linefeeds and level-three headings by 3 half linefeeds. These are absolute values and do not change if the line spacing changes from single to double or vice-versa.

To Define Processing of Paragraphs

!PARAGRAPH(1=s,2=i)

The beginning of a new paragraph is indicated to the Text Processor with the line layout command "p". The Text Processor will always ensure that at least the first three text lines of the paragraph appear together on a single page. Furthermore, it can be instructed by the PARAGRAPH command to precede each paragraph with s half linefeeds and to indent its first line by i spaces. The default value for s is the line spacing chosen (single or double spacing of lines), and for i an indention of zero spaces. With the PARAGRAPH command these defaults can be selectively modified.

```
!PARAGRAPH(1=5,2=3)
```

for instance, specifies that 5 half linefeeds are to be issued before each paragraph and that the first line of each paragraph is to be indented by three spaces, while

```
!PARAGRAPH(2=5)
```

would change only the indention to 5 spaces leaving the line spacing value unchanged.

To Set Tabs

!SETTAB(<num>=<pos>,....,<num>=<pos>)
Default: !SETTAB(1=5,2=10,...,14=70)

The use of tab positions greatly facilitates the composition of tabular information. The user can specify labelled tab positions using the "SETTAB" command. <num> specifies the label of the tab, an integer number between 1 and 100, while <pos> specifies the corresponding position of the tab, <pos> being an integer number between 1 and the selected width of the line. The Text Processor will clear all tabs that might have been set previously. It will restore the default tab setting with tab 1 at position 5, tab 2 at position 10, and so forth until tab 14 at position 70. It will then evaluate the parameter list of the "SETTAB" command and store the user-defined tab positions. These can override default tab positions (only specified tab positions are changed).

Example:

!SETTAB(1=17,2=36,3=59)

will set tabs in positions 17, 36 and 59 cancelling the tabs in positions 5, 10, 15 from the default setting leaving the others unaltered. The user now can use the "tab text" command described below to space text to the desired positions.

Note:

. These tabs should not be confused with the tabs set by the Text Editor for the on-line terminal.

To Define a Table.

!TABLE(<pos1>,<pos2>,...,<pos12>)

The commands described for pre-formatted and tabular text, like "SETTAB", "a", "tn", etc. allow the setting of tables containing short, well defined items like numbers, etc. The maximum number of columns that can be defined is six.

In general a table can be seen as a sequence of table entries, where each entry may contain one or more text columns, and each text column within an entry may contain one or more lines of text. The task to set such a table using the above commands becomes very elaborate, whenever text of variable length is to be set for a particular entry and column of a table. To ease this task, special table processing commands are provided.

The TABLE command allows the user to define a table containing up to six columns. For each column the beginning and ending print position must be specified. The minimum width for a particular table column is 8 print positions.

To ensure that table columns do not overlap, the Text Processor will sort the positions specified into ascending order and group them into pairs. The first pair now specifies column one, the second column two, and so forth.

To Generate Table of Contents

!CONTENT

The Text Processor can be instructed to prepare a table of contents for the text processed. The following elements specified in the text to be processed will constitute the table of contents:

. The width and depth in use at the occurrence of the "CONTENT" command will be maintained for the table of contents. The width must be at least 50 printpositions.

. The first title specified after the occurrence of the CONTENT command, but before the first heading, will be the title of the

table of contents. Titles following the first one will be ignored. If a heading occurs before the first title, all subsequent titles will be ignored. If no title is specified, the Text Processor will generate one.

. All headings encountered after the occurrence of the CONTENT command will be represented in the table of contents as follows:

  - Headings of level one and two will constitute the table of contents for the entire volume.

  - Headings of level one, two and three will constitute the table of contents for each chapter.

. The occurrence of a running footing indicated by the RFOOT command in the text will define the beginning of a new chapter.

. The running footing of the volume table of contents will be "Table of Contents", while those for the chapter table of contents will be as defined by the user in the RFOOT command starting the chapter. It is therefore essential that each new chapter be started in the text with an ARAB command, immediately followed by the new running footing. Both are preferably preceded by an "n" line-layout command requesting the start of a new page.

. The Text Processor can be instructed by a program call parameter to format only one table of contents for the entire volume containing all headings (see Program Call).

The pages of the table of contents will be numbered with Roman numerals. To avoid possible confusion, the user should abstain from using the ROMAN command simultaneously with the CONTENT command.

Commands to Layout Text

To Specify and Terminate a Heading

!hi <text> !o

The Text Processor can recognize up to three level of headings. Level-one and level-two headings appear in the table of contents for a Volume, while level-three headings, (together with level-one and level-two headings) will only appear in the table of contents for a chapter (see CONTENT command above). The layout of the table of contents is also dependent on the level of the heading. Level-one headings are printed without indention, while level-two headings are indented by three spaces and level-three headings by six spaces.

The line-layout command "hi" specifies the beginning of a heading, and the parameter i, with i= 1, 2, 3, defines this level. The command terminates the processing of the current line, if any, and starts a new heading. The heading will be preceded by the number of half linefeeds defined for this level of heading (see HEADING command above).

The number of half linefeeds for this particular heading can be altered by specifying additional half linefeeds using the "s" layout command described below together with the "hi" command. !s3h3 , for example, will precede the heading with additional 3 half linefeeds.

The text following the heading command constitutes the heading. A heading can extend over more than one line of text and might contain other layout commands to format it, such as indention, underlining, etc. The text of the heading is terminated by the line-layout command "o". This text will also appear in the table of contents in a position corresponding to the level number.

To Begin a New Page

!n

The "n" line-layout command causes the text following it to begin on a new page. The Text Processor will terminate the processing of the current line, if any, and format the text assembled so far, if any, as a part page.

To Specify the Beginning of a Paragraph

!p

The line-layout command "p" specifies the beginning of a paragraph. The command terminates the processing of the current line, if any, and starts a new paragraph. The first line of the paragraph will be preceded by the number of half lines and indented by the number of spaces requested with the last PARAGRAPH command.

Immediate and Hanging Indentions

!in
!jm

An "i" line-layout command starts a new line of text and causes the first character of the line to be indented a specified number of spaces from the current left margin. This is known as an immediate indention. The parameter n specifies the number of spaces the text line is to be indented. If n is specified as zero ("i0"), a new line is started with no indention. When n is omitted ("i"), the program uses the n value specified in the last "in" command.

A "j" line-layout command does not start a new line but it causes all subsequently formatted text lines to be indented a specified number of spaces from the current left margin. This is known as a hanging indention. The Text Processor continues to indent each line thereafter until it encounters the next layout command that starts a new line. The parameter m specifies the number of spaces the subsequent text lines are to be indented. If m is omitted ("j"), the last value of m specified in a "jm" command is used.

Note:

. The indention must never cause the length of the printable line to be less than 8 characters. That is, the line length, less the margin, less the indention, must not be less than 8 characters.

Beginning and Ending of a Keep

!k <text> !r

A "k" line-layout command starts a new line indicating the beginning of a section of text which cannot be divided between two pages and cannot be moved from its original location. This is known as a regular keep.

The "r" line-layout command ends the section of text started with the "k" command and causes a new line of text to be started.

Note:

. A keep cannot exceed the depth of the text area. The Text Processor releases the keep at the point where it exceeds the depth of the text area.

. Formatted and tabular text can be part of a keep.

To Insert Blank Half Lines

!sn

An "s" line-layout command causes the Text Processor to begin a new line of text and to issue a specified number of half linefeeds before it prints the text following the layout command. The parameter n specifies the number of half linefeeds to be issued. If "s0" is specified, the new line of text will be printed on top of the preceding line of text.

Under certain circumstances, skip lines are removed by the Text Processor. To ensure that it does not remove skip lines which allocate space for supplemental copy (e.g. charts and figures), the "s" layout command should be used with a "k" layout command as follows:

```
!k !s10 !r
```

thus declaring the allocated space to be part of a keep.

Note:

. The "s" line-layout command can also be used to space formatted and tabular text.

To Tab Text

!tn

The "t" line-layout command causes the text following it to be printed at the tab position specified. The parameter "n" specifies the label of the tab. The position corresponding to the label must have been specified in a preceding SETTAB command, or must be one of the default tab settings.

The "t" command can optionally be preceded by a period (".") causing periods to be printed in the tabbed-over space to the left of the text.

To Set a Table

!xn

The "x" line-layout command is used to set a table. The Text Processor will verify that a table has been defined using the TABLE command, else it will ignore the "x" command.

The parameter n specifies the column of the table to be set. The Text Processor will check that this column has been defined in the last TABLE command, and that the beginning and ending print position are within the usable width of the line as defined by the last MARGIN and LENGTH commands. It will furthermore check that the column contains at least 8 print positions. If any of these conditions are not fulfilled, the "x" command will be ignored.

The text following the "x" command will now be formatted using the column specifications as boundaries. The user can freely use all line-layout commands to achieve the desired layout of the table column. The "a" and "c" line-layout commands are restricted to the width of the table column. Care should be exercised when using the "t" line-layout command, since it is not restricted to the width of the table column.

The use of another "x" command terminates the formatting of the current column and selects a new column for formatting. The Text Processor will obey the following rules:

1) If the new column is positioned to the right of the current column, it will be formatted as a new column in the same table entry.

2) If the new column is positioned to the left of the current column, or "underneath" the current column, the Text Processor will begin a new table entry with the new column.

3) If the parameter "n" is zero ("0"), the Text Processor will terminate the current table entry and exit from table processing.

4) The following rules apply for the use of the "s" line-layout

command in combination with an "x" command:

- If table processing has not been started and a command combination such as "!s5x1" is issued, where the "s" command precedes the "x" command, the blank half lines requested will be issued in full and the first table entry will start with a new linefeed equal to the default spacing.
- If table processing has started and a command combination such as "!s5x2" is issued, where the "s" command precedes the "x" command to begin a new column, the blank lines requested will be issued in full as part of the preceding table column. The linefeeds preceding the new table column are calculated according to 1) and 2) above.
- If a command combination such as "!x2s5" is issued, where the "s" command follows the "x" command, the requested blank half lines will be issued together with the first text for the column entry. The proper column entry begin will be determined using 1) and 2) above.

There is no need for the user to specify keeps to ensure that table entries are not split between pages. Care should however be exercised to avoid table entries that are too large in depth. These can spoil the page layout. In such cases it is advisable, to split the entry, eventually by repeating the column specification.

The Appendix contains several examples to illustrate the use of table processing commands.

To Begin and End a Footnote

!f <text> !z

An "f" line-layout command 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 ensure that the footnote reference and the footnote itself will appear in the same page of text.

The text of the footnote may contain line-layout commands to format it. The footnote text is terminated by the "z" line-layout command.

To Print Superscripts and Subscripts

!1 !2 !3

In order to print text in the superscript position it must be preceded by a "1" word-layout command.

In order to print text in the subscript position it must be preceded by a "3" word-layout command.

A "2" word-layout command prints the text following it in the normal line position. The following examples illustrate the use of these word-layout commands:

1. To obtain the following mathematical expression:

   $(a + b)^2 = a^2 + 2ab + b^2,$

   one would enter

   (a + b)!12 = a!12 + 2ab + b!12.

2. Entering the following text as input:

   a!12!3ij!2b!1ij!32

   would result in the printing of

$a^{2}_{ij} b^{ij}_{2.}$

To Underline Text

!+ !-

Text to be underlined should be preceded with the word-layout command "+" and followed by the word-layout command "-".

To Print a Marker

!m

Occasionally one might want to print a marker in the left margin reserved for this purpose by the Text Processor (see MARGIN command). This allows flagging of text that is to be corrected by hand. The line-layout command "m" can be inserted anywhere in the text and causes an asterisk ("*") to be printed in the left margin.

To Force Additional Blanks

!

It is sometimes desirable to insert additional blanks into unformatted text which are not to be deleted by the Text Processor. Preceding these blanks with an exclamation mark will force the Text Processor to recognize them as special and not as word-terminating blanks. The expression S C O P E for instance is obtained by entering S! C! O! P! E in the input text.

To Print an Exclamation Mark

!!

Two successive exclamation marks force the Text Processor to print an exclamation mark in the text.

Printing Text "As Entered"

The "a" and "c" commands described below will switch the Text Processor from its principal mode of operation, the processing of unformatted text, to a more restricted mode, the processing of pre-formatted text. In this mode the text is printed exactly as it appears in the input stream controlled only by the overall size of the text area or table column if in table processing (see "TABLE" and "x" commands above). Left-hand and right-hand margin settings which might otherwise be in force are ignored. So are commands forcing indention of text or its justification. In this mode the carriage return, line feed and backspace control characters are honoured.

To Begin "As Is" Text

!a

The "a" line-layout command forces the start of a new line. The text following it will be printed as entered using the full width of the page or table column (see Table Processing). No deletion or insertion of blank spaces will take place. If the text started by this command is longer than the permitted width of the page or table column, the Text Processor will begin a new line with the word that did not fit to the previous line. The text started by this command ends at the next layout command that starts a new line.

Centering Text Lines.

!c

The "c" line-layout command causes the text following it to be centered. The text is terminated by the next layout command that starts a new line.

The command acts the same way as the "a" command with the added capability of centering the text within the full width of the page or table column (see Table Processing).

Ending Tabular Text

!u

The "u" line-layout command allows formatted and unformatted text to be printed on the same line. The "u" command is entered in the input immediately before the text which is to flow (unformatted text is said to "flow"). It indicates to the Text Processor that the tabular text has been processed and the Text Processor is to start formatting text in the regular manner. The "u" command does not start a new line,it simply overprints the last line formatted with an "a" or "c" line-layout command.

The following examples illustrate the use of the "u" command to combine tabular and flowing text.

1. To print "AUTHOR" in the left margin as shown in the example:

AUTHOR AUTHOR is a text processing system developed at CERN, the European Organization for Nuclear Research, Geneva, Switzerland .....

the following text is entered (the !SETTAB and !tn commands are explained above):

!MARGIN16 !SETTAB(1=4) !s3at1 AUTHOR !u AUTHOR is a text processing system developed at CERN, the European Organization for Nuclear Research, Geneva, Switzerland .....

2. To print "AUTHOR" in the right hand margin as shown in the example:

AUTHOR is a text processing system developed at CERN, the European Organization for Nuclear Research, Geneva, Switzerland ..... AUTHOR

the following text is entered:

!LENGTH54 !SETTAB(1=65) !MARGIN3 !s3at1 AUTHOR!u AUTHOR is a text processing system developed at CERN, the European Organization for Nuclear Research, Geneva, Switzerland .....

Grid Drawing Commands

Very often tables containing numbers and names have to be printed. Using the "a" and "t" commands or the table processing commands, it is easy to layout the data in the table. The grid necessary to separate the data could be drawn using the characters "-", "+" and "|" within "as is" lines. The slightest change of the layout of such a table however could cause a major editing job to adapt the old grid to the new layout. To eliminate this, the following grid drawing commands are made available:

To Begin a Grid

!b<list of positions>
!b,<list of positions>
!b

Any of the three forms of the "b" line-layout command will cause a new line of text to be started. It then causes a horizontal line to be printed starting at the leftmost position specified in the <list of positions> and ending at the rightmost position specified. The horizontal line occupies one text line. This command also starts a vertical line in each of the positions specified in the <list of positions>. These positions need not to be in ascending order. The vertical lines started

by this command continue until an "e" line-layout command is encountered.

The special form of the "b" command, "b,<list of positions>", adds the positions specified with this command to the ones specified by a previous "b" command, if any. The form "b" without <list of positions> uses the positions in use with the last "b" command, if any.

The <list of positions> contains the decimal numbers, separated by commas, that specify the print positions at which to start vertical lines. The list must be terminated by a blank. Printpositions that are beyond the current page width will be set to the page width.

To Draw a Horizontal Line

!l

The "l" line-layout command causes a new line of text to be started. It then draws a horizontal line between the leftmost and rightmost positions of the most recent "b" command. The intersections at the vertical lines are printed automatically by the Text Processor. The line occupies one text line.

To End a Grid

!e

The "e" line-layout command causes a new line of text to be started. It then causes a horizontal line to be drawn between the leftmost and the rightmost positions of the most recent "b" command and terminates all vertical lines started by this "b" command.

Vertical lines started by a "v" line-layout command are <u>not</u> terminated by an "e" command.

To Draw Line Segments

!g<list of positions>
!g,<list of positions>
!g

Any of the three forms of the "g" line-layout command causes a new line of text to be started. It then causes one or more horizontal line segments to be drawn as specified in the <list of positions>. If the "g" command is specified within a grid, the Text Processor will automatically print the intersections at the vertical lines.

The special form of the "g" command, "g,<list of positions>", adds the positions specified with this command to the ones specified by a previous "g" command, if any. The form "g" without <list of positions> uses the positions in use with the last "g" command, if any.

The <list of positions> contains decimal numbers, separated by commas, specifying the beginning end ending positions of the line segments to be drawn. The list must be terminated by a blank. The Text Processor will sort this list into ascending order and group them into pairs, each of them specifying a line segment.

To Begin Vertical Lines

!v<list of positions>
!v,<list of positions>
!v

Any of the three forms of the "v" line-layout command will cause a new line of text to be started. The "v" line-layout command starts a vertical line in <u>each</u> of the positions specified in the <list of positions>. These positions need not to be in ascending order. The vertical lines started by this command continue until a "q" line-layout command is encountered that cancels them.

The special form of the "v" command, "v,<list of positions>", adds the positions specified with this command to the ones left over from the previous "v" command, if any. The form "v" without <list of positions> uses the positions left over from the last "v" command, if any (see "q" command).

The <list of positions> contains the decimal numbers, separated by commas, that specify the print positions at which to start vertical lines. The list must be terminated by a blank. Print positions that are beyond the current page width will be set to the page width.

To Terminate Vertical Lines

```
!q<list of positions>
!q
```

None of the two forms of the "q" line-layout command cause a new line of text to be started. They merely terminate all or some of the vertical lines started by the last "v" command.

The "q" command without parameter list will terminate all vertical lines started by the last "v" command. It however does not remove them from the list of "v" positions. A subsequent "v" or "v,<list of positions>" command can still access them.

The "q<list of positions>" command will selectively terminate vertical lines in operation at these positions and remove them from the list of "v" positions.

The <list of positions> contain the decimal numbers, separated by commas, that specify the printpositions at which to terminate vertical lines. The list must be terminated by a blank.

Documentation of Programs

The Text Processor provides a secondary output file, called the code file, for the generation of source programs that are embedded in the input file of the Text Processor. A program and its documentation may thus be maintained as a single AUTHOR text file. The default name of the code file is "CODE". The program call statement allows to specify a different name (see Appendix: Program Call Conventions).

To Specify Code

```
!Cn

<any number of program statements that do not include the character
exclamation mark (!)>

<any line that contains an exclamation mark (!)>
```

The !Cn command must be the only information on an input line. The input lines following the !Cn command are treated as lines of code until a line includes an exclamation mark. Each code line is added to the formatted text, as if it were preceded by the the text processor commands !s2am; i.e. each code line is formatted as a separate line in "as is" mode and marked by an asterisk in the left margin. The first code line following the !Cn command is separated by n half-lines from the preceding text. For any small letter of a code line the corresponding capital letter will be printed.

Each code line is also converted to display code and written as a card image to the code file. The code file will thus be in the format of an input file to the CDC language processors UPDATE, FORTRAN, COMPASS, etc.

A code line that consists of the word

7-8-9

only generates a record mark on the code file.

The Reference Facility

When preparing scientific documentation one very often collects the references to be quoted separately and marks their quotations in the text symbolically. Once the documentation is ready the references are ordered and numbered. Their symbolic quotations in the text are then replaced by the number given to the reference.

The Text Processor offers the reference facility to ease this process. The user is advised to collect all the references for the paper he is preparing in a separate Section of his AUTHOR set.Using the Text Editor it is easy to maintain the intended order of the references. For processing of the text this Section must be at the beginning of the file presented to the Text Processor. It will take account of the references, number them and exchange their symbolic quotations in the text by these numbers. Having processed the entire text, the Text Processor will format and print the references.

If the Text Processor finds quotations in the text which have not been defined, it will flag them by printing an asterisk in the margin.It will also flag references that have been defined but not quoted in the same manner.

To Define a Reference and its Symbol

!REFERENCE,[symbol]=<text of reference>!EREFERENCE

where <symbol> can be any collection of alphanumeric characters up to seven characters. The <text of reference> may contain word-layout commands such as "!+", "!-", "!1" etc. It should however not contain line-layout commands since these might conflict with the ones used by the Text Processor to format the references.

To Quote References

![<symbol>]
![<symbol>,...,<symbol>]

where <symbol> is the symbol used with the definition of the reference to be quoted. Several such <symbol> might be quoted simultaneously. They must be separated by commas.

AUTHOR

Reference Manual

Appendix

AUTHOR System Command Summary:

Command Form	Description and Sample Command
File Manager Commands	
ACCESS,<pf name> [,<p list>].	to access an AUTHOR set ACCESS,manual.
CREATE,<pf name> [,p list>].	to create an AUTHOR set CREATE,manual.
ALTER,{<lfn>/<pfn>} [,<p.list>].	SCOPE perm. file command
ATTACH,{<lfn>/<pfn>} [,<p.list>].	
CATALOG,{<lfn>/<pfn>} [,<p.list>].	
EXTEND,{<lfn>/<pfn>} [,<p.list>].	
GETPF,{<lfn>/<pfn>} [,<p.list>].	
PURGE,{<lfn>/<pfn>} [,<p.list>].	
RENAME,{<lfn>/<pfn>} [,<p.list>].	
SAVEPF,{<lfn>/<pfn>} [,<p.list>].	
RETURN,<list of files>.	SCOPE file command
REWIND,<list of files>.	
ADD,<entry name>.	to add a Section ADD,s=New Section.
DELETE,<entry>, E=<edition>	to delete an entry DELETE,#3,E=2.
BEFORE,<entry number>, <entry name>.	to insert a section before an existing entry BEFORE,#2,C=Abstract,S=1.
AFTER,<entry number>, <entry name>.	to insert a section after an existing entry AFTER,#2,C=Format Specifications,S=1.
NAME,<entry number>, <entry name>.	to change the name of an existing entry NAME,#1,V=Author System.
DIRECTORY[,<entry>] [,E=<edition>].	to display all or part of the File Directory DIRECTORY,#4,E=*.
PROOF[,<entry>] [,E=<edition>][,<file>].	to obtain a proof copy PROOF,#4.

AUTHOR

Reference Manual

Appendix

AUTHOR System Command Summary:

Command Form	Description and Sample Command
READ,<entry>[,<file>].	to read information into a section READ,S=1,D=compile.
WRITE[,<entry>] [,E=<edition>][,<file>].	to write information from a section WRITE,#1,E=1,a=text.
EDIT[,<entry>] [,E=<edition>].	to edit a section EDIT,#3,E=1.
BRIEF. VERBOSE.	to reduce amount of prompting to obtain full prompting
#n #	to control the field length #25600
Text Editor Commands	
A	to abort editing
Cs	to specify s as universal character where s can be any non-alphanumeric character default is *
C	specify no universal character
E	to end editing
E+ E-	end editing of section, continue with next (or previous) section
J	jettison the effect of the last command line
Kd	retain current position in cursor slot d, d = 0, 1, ..., 9
Ln	loop, executing remainder of command line n times
L	Loop, executing remainder of command line until end (or beginning) of text
T,n_1,n_2, ..	set tabs at n_1, n_2, ... default is a tab at 7
T	set tabs at 11, 20 and 36
Ut	retain t as default unit of text with t=C,W,L default is Character
Vnt	display window of n units of text
V	suppress display of verification window
Sp	skip text until position p s5 skip 5 characters s-5 go back 5 characters s2w skip 2 words s3l skip 3 lines s/HvE/ search string HvE
Dp	delete text until position p De delete until end of text
Pp	print text until position p Pb print from beginning
I	begin input from terminal
I/new/	insert new string
Rit/old/new/	replace old string by new string i times R-5/Hve/HvE/
I(f)	insert text from file I(-file)

AUTHOR

Reference Manual

Appendix

AUTHOR System Command Summary:

Command Form	Description and Sample Command
W(f)p	write text until position p on file
D(f)p	delete text until position p and save on file
	D(-text)e
#n #	reset or request field length
Text Processor Commands	
Page Shaping	
!WIDTHw	width of page, default is 70 characters
!DEPTHd	depth of page, default is 120 half lines
	minimum is 25 half lines
!MARGINm	left margin, default is 0
!LENGTHl	line length, default is the minimum of 70
	or width
	the absolute minimum is 8
!SPACINGs	line spacing, default is 2 half lines
!SINGLE	s=1,...,10 is possible
!DOUBLE	
!COLJUST	text area justification
!ECOLJUST	
Page Framing	
!TITLE <text> !ETITLE	specify title
!GAPg	change gap default to g=0,...,10
	default is a gap of four half lines
!ARAB(xxxxx)	restart page numbering
!ROMAN	restart page numbering (I - X)
!RFOOT <text> !	running footing
!DATE	select or deselect printing of date
!EDATE	default is printing of date
!FRONT	print only front pages
!EFRONT	return to recto/verso printing (default)
Text Shaping	
!JUST	line justification
!EJUST	
!HEADING(1=i,2=j,3=k)	linefeeds for headings
!PARAGRAPH(1=s,2=i)	s half linefeeds; i indention
!SETTAB(<num>=<pos>,...)	set tabs, default is 1=5,...,14=70
!TABLE(<pos1>,...,<pos12>)	define a table, up to six columns
!CONTENT	generate table of contents
Text Layout	
!hi <text> !o	specify level and text of heading
!n	begin a new page
!p	begin a new paragraph
!i0	begin a new line with zero indention
!in	begin new line with "n" spaces indention
!jn	hanging indention
!k <text> !r	keep together
!s0	print on top of previous line
!sn	insert n half lines
!tn	tab to position n
!x0	terminate table processing
!xn	set column n of table

AUTHOR

Reference Manual

Appendix

AUTHOR System Command Summary:

Command Form	Description and Sample Command
!f <text> !z	footnote text
!1 !2 !3	super script, normal line, subscript
!+ !-	begin and end underlining
!m	print a marker
!!	print an exclamation mark
!	add additional blanks
Text As Entered	
!a	begin of "as is" text
!c	center text lines
!u	overprint "as is" text
Grid Drawing	
!bn,..,m	begin a grid at positions n,...,m
!b,n,...,m	
!b	
!l	draw a line through grid
!e	terminate grid drawing
!gn1,n2,...,m1,m2	draw line segments
!g,n1,n2,...,m1,m2	
!g	
!vn,...,m	begin vertical lines at n,...,m
!v,n,...,m !v	
!qn,...,m	selectively terminate or terminate all
!q	vertical lines started with last v command
Program Documentation	
!Cn	begin of code lines, skip n half lines
7-8-9	generate record mark on code file
Reference Facility	
!REFERENCE,[<symbol>]= <text of reference> !EREFERENCE	define reference symbol and its text
![<symbol>,...,<symbol>]	quote one or more references in text

AUTHOR

Reference Manual

Appendix

Text Processing Examples

This Appendix contains several examples to format text with the Text Processor. The formatted text and the input text are both printed. The input text has been reproduced with a margin of eight characters. Whenever additional explanations have been inserted, a margin of three characters has been used. In addition the beginning and ending of such explanatory text has been indicated by a star ("*") in column one.

Page Shaping and Framing

This more general layout example contains the input text necessary to produce the first and part of the second page of this manual. It illustrates the use of page shaping and framing commands as well as those necessary to produce headings, paragraphs and blocks of text.

Input Text:

```
!WIDTH76 !LENGTH76 !DEPTH158
!COLJUST !HEADING(1=8,2=5,3=4) !PARAGRAPH(1=3,2=0)
!SETTAB(1=47,2=54)
!a
!s20t1 CERN - Data Handling Division
!t2 DD/76/22
!t2 REVISED
!s3t2 Horst von Eicken
!t2 November 1977
!s25c A U T H O R
!s4c a
!s4c Text Processing System
```

* The front page has been formatted using the text "as entered" facility. The page width specified is larger than the default of 70 characters per line. The "LENGTH" command must therefore be used, to obtain the full width for all lines. In addition please note that a first blank line has been specified before the initial skip of 20 half lines was given. Without this blank line, the Text Processor would have ignored the initial skip since we are at the beginning of a page. This technique works, if one knows that a new page is started. In all other cases, when one wants to reserve space for a figure for instance, it is necessary to
* specify a keep around the "s" command ("!k !s10 !r").

```
!n !CONTENT !ARAB
!TITLE !c AUTHOR !s3c Reference Manual !ETITLE
!RFOOT Introduction !
!h1 !+INTRODUCTION!- !o

!ps2 AUTHOR is an interactive text-processing system for Control
Data 6000/CYBER computer systems. It has been designed to aid
authors, editors, and secretaries to produce reports, letters and
other documents.

!p The following major subsystems are contained in the AUTHOR
text-processing system:

!a     FILE MANAGER !ui20j20 to manage a random-access file
containing the text of one or more Volumes separated into Chapters

!a     TEXT EDITOR !ui20j20 to edit a Section extracted by the FILE
MANAGER;

!a     TEXT PROCESSOR !ui20j20 to process information prepared by
the TEXT EDITOR and extracted by the FILE MANAGER,
```

!a POST PROCESSORS !ui20j20 to adapt information processed by the TEXT PROCESSOR and extracted by the FILE MANAGER

* The command combination "!ps2 " has been used to begin the first paragraph after the heading of level one. Since we defined with the PARAGRAPH command, that each paragraph should be preceded by three half lines, we effectively begin this paragraph with five half lines. In other words, line skips are added together.

The above text also contains the use of the "u" line-layout command to obtain a "labelled" block of text. The subsequent example on "immediate
* and hanging indentions" shows other ways to obtain similar layouts.

Immediate and Hanging Indentions

The insert blank half lines command "s" together with the immediate and hanging indentions command "i" and "j" are very useful to obtain "labelled" blocks of text. Consider the following formatted text:

I) Let us assume we have to print some blocks of text which are to be labelled with the Roman numerals I through V, where each numeral is to be followed by a closing parenthesis ")" and a blank space separating it from the text of the first line of the block. The "largest" numeral, namely "III) ", will occupy five print positions. Let us furthermore assume, that the numerals should be indented by three print positions, then the command combination "!s3i3j8 " would cause such a text block to be started preceded by three blank half lines.

II) Since our first numeral "I) " occupies only three print positions and since we want the numerals to immediately precede the text, we will start our first block of text with the command combination "!s3i5j8 ", the second block with "!s3i4j ", the third block with "!s3i3j " and so forth.

III) Since we are never changing the value of the hanging indention "j" after we have defined it for the first block, we don't have to repeat its value. It is sufficient to just state the "j" again.

IV) When adjusting text lines to be flush with the left- and right-hand margin the text processor will insert additional blank spaces between the words of a line. In order to avoid such an insertion between the label of a block and its first text word, we have to declare the blank separating the label from the text to be a "special" blank by preceeding it with an exclamation mark. The text string "!s3i4j IV)! When" was for instance used to begin the current block of text.

V) The "u" command to end tabular text and the "x" command to format table entries offer additional possibilities to "label" blocks of text.

The input text to format the first block is:

```
!s4i5j8 I)! Let us assume we have to print some blocks of text which
are to be labelled with the Roman numerals I through V, where each
numeral is to be followed by a closing parenthesis ")" and a blank
space separating it from the text of the first line of the block.
The "largest" numeral, namely "III) ", will occupy five print
positions. Let us furthermore assume, that the numerals should be
indented by three print positions, then the command combination
"!!s3i3j8! " would cause such a text block to be started preceded by
three blank half lines.
```

Tab Setting and Grid Drawing

A table containing numerals and short text is to be formatted. The SETTAB command in conjunction with grid drawing can be used as follows:

Input Text:

```
!SETTAB(1=14,2=16,3=31,4=41,5=53)
!k
!b12,29,39,63
!at1 SHARE !t3 PRICE !t4 PRICE RANGE ADJUSTED
!b,51
!s4at1 !+Switzerland!- SFr !t3 15.9.77 !t4 1973-1976 !t5    1977
!g29,63
!s2at2 Alusuisse I !t3  1535 !t4 2167- 870 !t5 1620-1350
!at2 Alusuisse N !t3   650 !t4 1023- 340 !t5  675- 515
!at2 Nestle I !t3  3590 !t4 4338-2050 !t5 3605-3280
!at2 Nestle N !t3  2215 !t4 2572-1200 !t5 2235-1985
!l
!s4at1 !+CANADA!-    Can.$ !t3 15.9.77 !t4 1973-1976 !t5    1977
!g29,63
!s4at2 Bell Canada !t3    54 !t4   52-  39 !t5   56-  40
!at2 Shell Canada !t3    16 !t4   22-  10 !t5   17-  13
!a  !er
```

* Please note, that the keep command "k" has been used together with the command to release the keep "r". This makes shure, that the table is not
* split between pages.

Processed Text:

SHARE	PRICE	PRICE RANGE ADJUSTED	
Switzerland SFr	15.9.77	1973-1976	1977
Alusuisse I	1535	2167- 870	1620-1350
Alusuisse N	650	1023- 340	675- 515
Nestle I	3590	4338-2050	3605-3280
Nestle N	2215	2572-1200	2235-1985
CANADA Can.$	15.9.77	1973-1976	1977
Bell Canada	54	52- 39	56- 40
Shell Canada	16	22- 10	17- 13

Table Processing

The table processing commands TABLE and "x" allow the formatting of a table with up to six columns. It should be used, if the table entries are not just numbers or short text. The subsequent example illustrates the combined use of table processing and grid drawing commands.

Input Text:

```
!TABLE(13,22,25,36,39,70)
!k !b12,23,37,72
!s2x1c EQUIPMENT !s2c ERROR #
```

AUTHOR

Reference Manual

Appendix

```
!x2c DATA !s2c MODULE
!x3c DESCRIPTION
!l
!s2x1c 1 !x2c MOTOR !c PUMP !c PMT
!c XWCD !c XWCA
!x3 Time - out from Analogue Acquisition. This fault occurs !+only!-
on reading, and does not affect the control in any way.
!x0s2g23,72
!s2x2c MAG !x3 Time-Out. The Siemens Computer did not respond within
1 second.
!l
!s2x1c 2 !x2c MAG !x3 Supply fail. !l
!s2x1c 3 !x2c MAG !x3 Magnet fail. !l
!s2x1c 4 !x2c MAG !x3 Unstable. !l
!s2x1c 5 !x2c MAG !x3 Busy. !l
!s2x1c 6 !x2c MAG !x3 Request Error. This error occurs when
something goes wrong with the dialogue between the two computers. !l
!s2x1c 7 !x2c MAG !x3 Attempt to control a circuit breaker on a
magnet which does not have one.
!l
!s2x1c 19 !x2c All !x3 !+Permission Error!-. An attempt has been
made to exercise a control restricted to the EA group without having
the EA group password.
!e !x0 !r
```

* Please note that the entire table has been surrounded by a keep command.
* Thus the table will not be split between pages.

AUTHOR

Reference Manual

Appendix

Processed Text:

EQUIPMENT ERROR #	DATA MODULE	DESCRIPTION
1	MOTOR PUMP PMT XWCD XWCA	Time - out from Analogue Acquisition. This fault occurs only on reading, and does not affect the control in any way.
	MAG	Time-Out. The Siemens Computer did not respond within 1 second.
2	MAG	Supply fail.
3	MAG	Magnet fail.
4	MAG	Unstable.
5	MAG	Busy.
6	MAG	Request Error. This error occurs when something goes wrong with the dialogue between the two computers.
7	MAG	Attempt to control a circuit breaker on a magnet which does not have one.
19	All	Permission Error. An attempt has been made to exercise a control restricted to the EA group without having the EA group password.

AUTHOR

Reference Manual

Appendix

Program Documentation

The subsequent example wants to give an impression on how one can use the Text Processor in order to facilitate the task of program documentation. Please note that the code lines in the processed text are flagged by an asterisk in the margin. In addition they will be written in display code to a user specified file ready for processing by CDC language processors such as UPDATE, FORTRAN, COMPASS, etc. (see Appendix: Program Call Conventions)

Input Text

```
      Terminal Output Routine
      !p The terminal output routine "TERM! OUT" is used by AUTHOR to
      output processed text to the on-line user terminal. It is the task
      of this subroutine to translate the control codes and characters
      of the AUTHOR character set into the corresponding, device dependant
      codes. The subroutine "TTY! OUT" is then finally used to perform
      the actual transfer of the device dependant code string.

      !C3
            SUBROUTINE TERM OUT ( text, length )
      !p The subroutine "TERM! OUT" has two parameters:
      !TABLE(1,10,15,76)
      !s3x1 TEXT !x2 a one dimensional array containing the text to
      be translated by TERM! OUT. The text is stored one code per word,
      rightadjusted, zero fill.
      !s1x1 LENGTH !x2 the size of the text string stored in TEXT
      !p The common block /TERMCOM/ is used for communication between
      "TERM! OUT" and the terminal dependant subroutines.

      !C3
            COMMON /TERMCOM/ TERMLST  (8), TERMNAL, SPACE (7), FEED (7),
           1                 SLEVEL (7,7), TSCRIPT
      !s3x1 TERMLST !x2 a one dimensional array containing the names
      of the various terminals supported by AUTHOR. The terminal names
      are stored in display code, one name per word, left adjusted,
      zero fill.
      !s1x1 TERMNAL !x2 an index value indicating the terminal in use.
      !s1x1 SPACE !x2 a one dimensional array containing the width of the
      characters in relative half units for each supported terminal.
      !s1x1 FEED !x2 a one dimensional array containing the leading value
      in relative half units for each supported terminal.
      !s1x1 SLEVEL !x2 a two dimensional array containing the deviation
      for
      the six script levels from the base line and the base line itself
      for each supported terminal. !i5 SLEVEL ( SCRIPT, TERMNAL )
      !s1x1 TSCRIPT !x2 an index value indicating the current script level
      with !i5 1 ! ! being the super-superscript,
      !i5 4 ! ! being the base line,
      !i5 7 ! ! being the sub-subscript level. !x0
```

Processed Text:

Terminal Output Routine

The terminal output routine "TERM OUT" is used by AUTHOR to output processed text to the on-line user terminal. It is the task of this subroutine to translate the control codes and characters of the AUTHOR character set into the corresponding, device dependant codes. The subroutine "TTY OUT" is then finally used to perform the actual transfer of the device dependant code string.

```
*       SUBROUTINE TERM OUT ( TEXT, LENGTH )
```

The subroutine "TERM OUT" has two parameters:

TEXT	a one dimensional array containing the text to be translated by TERM OUT. The text is stored one code per word, rightadjusted, zero fill.
LENGTH	the size of the text string stored in TEXT

The common block /TERMCOM/ is used for communication between "TERM! OUT" and the terminal dependant subroutines.

```
*       COMMON /TERMCOM/ TERMLST  (8), TERMNAL, SPACE (7), FEED (7),
*      1                 SLEVEL (7,7), TSCRIPT
```

TERMLST	a one dimensional array containing the names of the various terminals supported by AUTHOR. The terminal names are stored in display code, one name per word, left adjusted, zero fill.
TERMNAL	an index value indicating the terminal in use.
SPACE	a one dimensional array containing the width of the characters in relative half units for each supported terminal.
FEED	a one dimensional array containing the leading value in relative half units for each supported terminal.
SLEVEL	a two dimensional array containing the deviation for the six script levels from the base line and the base line itself for each supported terminal. SLEVEL (SCRIPT, TERMNAL)
TSCRIPT	an index value indicating the current script level with 1 being the super-superscript, 4 being the base line, 7 being the sub-subscript level.

AUTHOR

Reference Manual

Appendix

Program Call Conventions

AUTHOR

AUTHOR has been implemented at CERN under the SCOPE 3.4.3 Operating System with INTERCOM 4.3 as interactive sub-system. AUTHOR can be attached and called as follows:

```
ATTACH,AUTHOR,ID=PROGLIB.          or  FETCH,AUTHOR,PROGLIB.
AUTHOR[,user][,permanent file id].
```

where "user" is the user's name, up to seven characters long, and "permanent file id" is his identification for the SCOPE permanent file system. Both parameters are optional.

CERNTXT

The stand-alone Text Processor CERNTEXT can be attached and called as follows:

```
ATTACH,CERNTXT,ID=PROGLIB.          or  FETCH,CERNTXT,PROGLIB.
CERNTXT,TEXTIN,TEXTOUT,BELL,CODE,CONTENT.
```

where the parameters have the following meaning:

TEXTIN — the file containing the text to be processed by CERNTXT. TEXTIN is assumed to be a disk file and will be rewound prior to its usage. The following three special file names can be used :

```
TTY
HPLEFT
HPRIGHT
```

The file name TTY directs CERNTXT to read from the keyboard of the user terminal. In order to terminate the input, the user must send an end-of-file from the terminal. (A line containing nothing else but %EOF will perform this, if the terminal is connected via SUPERMUX.)

The two file names HPLEFT and HPRIGHT can be used for the HEWLETT PACKARD Mini Data Station 2644 A to read the left hand or right hand cassette unit.

TEXTOUT — The file TEXTOUT is assumed to be a disk file and will contain the processed text. It is rewound prior to usage. Please note that the file produced cannot be transferred to any device by SCOPE commands, it has to be post processed by SHOWIT for the desired output device.

The following special file names can be used, to direct the processed text directly to the on-line terminal:

```
DTC
DTCS
DIABLO
```

for the three types of hard copy terminals available, and

```
SCREEN
T4012
```

for all display terminals. (T4012 addresses either the TEKTRONIX T4012 or the T4006 display.)

If the special file name "COMPILE" is used, CERNTXT will not produce any text output, but merily extract from the file TEXTIN the code lines and copy them to the file "CODE" ready for compilation.

BELL — This parameter, if specified with "BELL", will cause CERNTXT to ring the bell at the terminal , stop the output and wait for a carriage return, given by the user, every time a form feed has been issued. This allows the user to mount special paper in the hard copy terminals to obtain for instance the printout on pre-printed letters, forms or the special paper to be used for the printing shop at CERN. This parameter should not be specified, if the processed text is directed to a disk file.

CODE — This parameter specifies the file name to be used for the display code file containing the code to be compiled or assembled. (See the "!C" command under "Documentation of Programs" for details.)

CONTENT — The CONTENT parameter directs the processing of the table of contents, if one has been requested. If "FULL" is specified as actual parameter, CERNTXT will produce only one table of contents for the entire text processed. This table of contents will contain all levels of heading.

Note:

AUTHOR uses the "Rubout" key to delete the last input character. The "Backspace" key will backspace one character, but not delete it.

AUTHOR

Reference Manual

Appendix

Availability of AUTHOR

Program Availability and Charging

AUTHOR is part of the CERN Program Library and may be used freely inside CERN. Universities, national laboratories and other non-profit organizations in Member States may receive AUTHOR and its documentation freely and without charge, subject only to a limitation on the number of copies to be sent to any one institution free of charge.

Other organizations (commercial firms) are expected to pay a small fee to cover our immediate expenses in supplying the desired material. This fee is 150.-- SFrs. for AUTHOR and includes the cost of a magnetic tape.

Universities and other non-profit organizations in non Member States normally pay a small fee as in the preceding paragraph, but the fee is waived in the case of laboratories collaborating on CERN experiments or engaged in a reciprocal exchange agreement with CERN.

CERN reserves the right to charge a higher fee, up to the full cost of developing the programs, or to refuse any request which it deems to be not in the interest of the Organization.

Conditions of Use

Programs and documentation are provided solely for the use of the organization to which they are distributed and may not be redistributed to any third party without the express agreement of CERN.

The material cannot be sold.

CERN should be given credit in all references, library documentation, and publications based on the programs.

CERN undertakes no obligation for maintenance of the programs, nor responsibility for their correctness, and accepts no liability whatsoever resulting from the use of its programs. Although we do not "support" the programs in a commercial sense, we will answer questions concerning the implementation or usage of the programs and we welcome suggestions for improvements.

Requests for AUTHOR should be addressed to:

The Program Librarian
Data Handling Division
CERN

CH 1211 GENEVA 23
SWITZERLAND